universität wien

MASTERARBEIT

Titel der Masterarbeit

Detecting sets of linked key players in social networks

Verfasser

Marlene Weiss, B.Sc.

angestrebter akademischer Grad

Diplom-Ingenieur (Dipl.-Ing.)

Wien, Oktober 2012

Studienkennzahl lt. Studienblatt:	A 066 926
Studienrichtung lt. Studienblatt:	Masterstudium Wirtschaftsinformatik
Betreuer:	Univ.-Prof. Dr. Walter J. Gutjahr

Abstract

In *Social Network Analysis* (SNA) the concept of *Group Betweenness Centrality* (GBC) is a unit to measure the influence of a group within a network. It is defined as the more shortest paths of the network pass through a subset of individuals, the greater the betweenness of this subset of individuals is.

There are algorithms for determining the Group Betweenness Centrality and also for determining a subset of given size with maximum GBC. However, the aim of this thesis is not only to find a group with maximum GBC, but also to develop an algorithm for finding a group in which every member is connected to every other member (also called clique).

As the problem itself is np-hard the proposed algorithm is of a meta heuristic nature. For quality assessment not only one algorithm was implemented, instead the two techniques *Simulated Annealing* (SA) and *Genetic Algorithm* (GA) were applied and compared.

Since most of the generated solutions are not feasible in the context of this problem statement, i.e. don't compose a clique, a suitable approach to either eliminate unacceptable solutions (*penalty-function method*) or only generate feasible solutions (*repair function method*), is required. The final algorithms work with two different penalty method approaches.

The underlying data used in the analysis is derived from real-world data sets of social networks as well as from synthetically generated data.

Contents

List of Figures

List of Tables

1 Introduction

„Society does not consist of individuals, but rather expresses the sum of relations in which these individuals stand to each other."
Karl Marx: A Contribution to the Critique of Political Economy, 1859

1.1 Problem statement

Every day tremendous amounts of data are produced and stored in the age of social media, social networks and the social web. It is impossible to extract information from this material manually, but it offers an excellent data base to analyse social behaviour with *Social Network Analysis* (SNA).

Communication nowadays depends heavily on emails, chats, phone calls and messages in online networks. Of higher interest to analyse are relationship networks which either can be as simple as "who knows whom" or even more complex by refining relationship with attributes, such as the intensity of relationships and the degree of kinship. All this data can be used to obtain knowledge and derive models that can explain group dynamics, how contagious disease spreads, how information is propagated, communication behaviour of mobile phone customers and even how terrorist organizations operate [9].

Generally within networks there are several measurements to describe their characteristics. Examples thereof are density, closeness and centrality. To estimate the potential monitoring and control capabilities a vertex may have on data flowing in a network [26] *Betweenness Centrality (BC)* is a measurement of utter importance.

Group Betweenness Centrality (GBC) is an enhanced concept of Betweenness Centrality. GBC can be used to estimate the influence of a group of vertices over the information flow in the network. It was first defined by *Everett* and *Borgatti* [12].

Puzis, *Elovici* and *Dolev* proposed an algorithm for the fast computation of Group Betweenness Centrality. Their approach reduces the calculation effort dramatically [26].

In 2006 *Borgatti* showed how to identify sets of key vertices in social networks, though these vertices were not necessarily connected with each other. Moreover,

Borgatti did not use Betweenness Centrality to evaluate the importance of a vertice [7].

In 2010 *Fink* and *Spoerhase* introduced the term *Maximum Betweenness Centrality* problem (MBC), which is defined as the search for a vertice with maximum GBC in a group C [14].

1.2 Objectives

The aim of this thesis is to investigate a modification of the topics researched by *Borgatti* (2006) and *Fink/Spoerhase* (2010), which arises from the additional constraint that the selected subset must be a clique, viz. each person of the subset must be connected to every other. This problem is hereby defined as *Maximum Group Betweenness Centrality* problem. (MGBC).

As the problem itself is np-hard, the proposed algorithm is of a meta heuristic nature. For quality assessment, the two different approaches *Simulated Annealing* (SA) and *Genetic Algorithm* (GA) will be implemented and compared.

Since most of the detected solutions will be infeasible, i.e. don't compose a clique, a suitable approach to eliminate unacceptable solutions (*penalty-function method*) is required. For both algorithm possible penalty methods will be analysed and two implemented for each.

In addition test data is essential for testing purposes. Not only will synthetically generated graphs be used as part of this thesis, but real world examples will also serve as models for abstracting information.

2 Fundamentals and Basic Principles

This chapter gives an introduction to networks in general and to Social Network Analysis. Topics covered include definitions, ancestry and forms of representation.

2.1 Networks in General

A network N consists of a set of vertices V and a set of edges E. Colloquially vertices can be called actors or nodes, and edges know as ties or connections. If one ore more nodes in a network cannot be reached either directly or indirectly, such a network is called disconnected. Furthermore such a network is divided into fragments, also called components. In this thesis only connected networks are given consideration.

Actor

In social networks a vertice is called an actor and can be represented by any social single person or group. Actors are discrete, individual, corporate, or collective social units [35]. These actors do not necessarily have to be humans. It is conceivable that they represent animals or groups of social actors. If an analysis focuses on actors of the same type, e.g. students of a specific university, these networks are called one-mode networks. However, it is also possible to create two-mode or even three-mode networks with different kind of actors. In this paper only one-mode networks are taken into account.

Tie

A tie defines the relationship between two actors. As shown by Wasserman in [35] the range of ties can be quite extensive:

- Individual evaluations (friendship, attraction, dislike)
- Transfer of material ressources (trade, business transaction)
- Transfer of non-material ressources (communciation, messages, money)
- Association or affiliation (belonging to the same group)

- Movement between places (migration, numbers of travellers)
- Kinship (descent, marriage)
- Physical connection (road, river, air connection)

Following [35] a tie can be refined by two properties; whether the relation is *directional* or *non-directional*, and whether it is *dichotomous* or *valued*.

In a *directional* connection a tie has a source and a destination. For example a person is attracted to another but this attraction is not returned. Within a *non-directional* connection a tie does not have such a direction. An example thereof is a marriage: Both partners are married to each other.

The second important property is whether a relation is *dichotomous* or *valued*. A dichotomous tie can be either be present or absent. This means two persons are either in a committed relationship or they are not. In contrast a valued tie can only be absent and present. Furthermore, if present, it provides information about the quality of the relationship. In a marriage for example, a valued tie can provide information as to how many years the connection has existent.

2.2 Representation of networks

Previously it was defined what networks are, what elements they consist of and which properties can refine such elements. Another important aspect of networks is their representation. For the purpose of this thesis two notations will be presented: the graph theoretic and the sociometric approach.

2.2.1 Graph theoretic

The origins of graph theory go way back to 1736 when Swiss mathematician Leonhard Euler published a solution to the *Königsberg bridge problem*. The question was whether there was a tour of the city of Königsberg (now Kaliningrad, Russia), which used every of the seven bridges over the river Pregel only once. Euler could prove with help of graph theory that such a tour does not exist [22].

In graph theoretic notation actors are represented by a geometrical forms, usually circles. Ties are always represented by lines between actors. Additionally there must be a distinction made between the visualisation of tie properties. Directional relations have an extra arrow specifying the direction of the the tie, valued relations have an axillary writing above their line. Figure 2.1 shows how the properties of ties are commonly pictured.

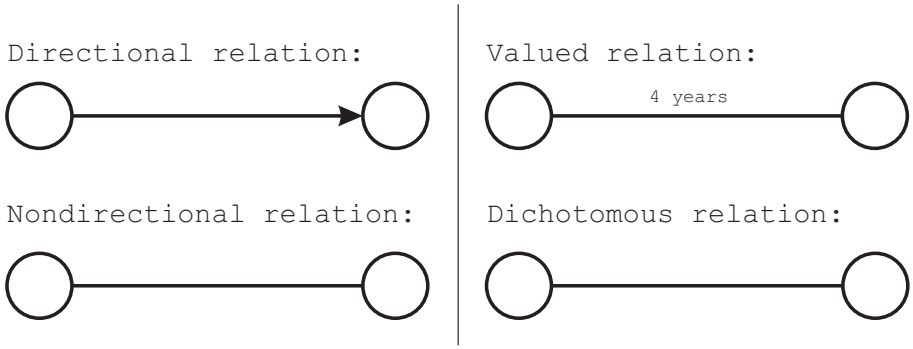

Figure 2.1: Tie properties in graph theoretical notation

To show how whole networks are drawn in graph theoretic notation, an example network N is taken, consisting of 5 vertices, $N = a, b, c, d, e$. Furthermore the network is non-directional and dichotomous. Figure 2.2 shows this example in graph theoretical representation.

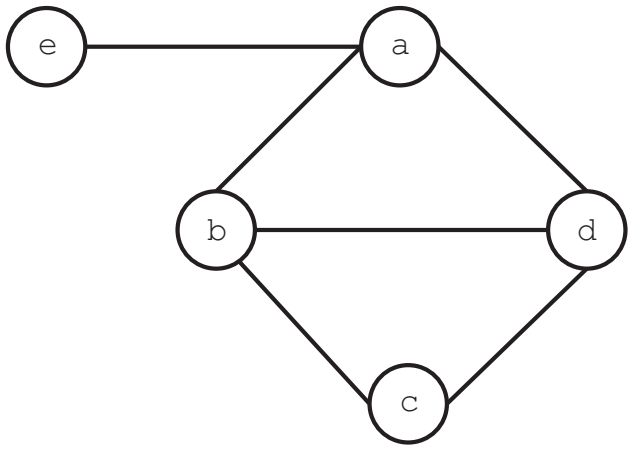

Figure 2.2: Network N in graph theoretical notation

2.2.2 Sociometric

As will be explained in section 2.3.2, the sociometric approach was introduced in the early 1930s. It mainly makes use of matrices where actors build the columns and rows. Ties are represented by the quantifiers within the matrix. If a binary notation is chosen (only 1 and 0 values are within the matrix, the network is dichotomous, otherwise valued. The directional property is realised by the mirror-inverted character of the matrix. With the help of the vertices α and β, figure 2.3 shows how properties are represented.

Directional relation:

	α	β
α	-	1
β	0	-

Valued relation:

	α	β
α	-	5
β	1	-

Nondirectional relation:

	α	β
α	-	1
β	1	-

Dichotomous relation:

	α	β
α	-	1
β	1	-

Figure 2.3: Tie properties in sociometric notation

Again the same example network N with vertices $N = a, b, c, d, e$ is assumed. Figure 2.4 shows the example again, this time sociometric representation

N	a	b	c	d	e
a	-	1	0	1	1
b	1	-	1	1	0
c	0	1	-	1	0
d	1	1	1	-	0
e	1	0	0	0	-

Figure 2.4: Network N in sociometric notation

2.3 Social Network Analysis

2.3.1 Definition

Social Network Analysis (SNA) can be seen as a systematic method for obtaining insights into a network of social actors and their relationships.

Based on the research of *Valdis Krebs*[1] *Jamali* and *Abolhassani* defined SNA as the mapping and measuring of relationships and flows between people, groups, organisations, animals, computers or other information/knowledge processing entities. They also stated that Social Network Analysis provides both a visual and a mathematical analysis of human relationships [19].

Borgatti even claims in [8] that SNA provides an answer to a question that has preoccupied social philosophy since the time of Plato, namely, the problem of social order: how autonomous individuals can combine to create enduring, functioning societies.

Freeman defines SNA as an approach used in the social sciences to describe the social structures of personal interactions that exist within a network [15]. In addition he defined four features for characterising SNA [15]:

- Structural intuition based on ties linking social actors
- Based on systematic empirical data
- Makes heavy use on graphic imagery
- Relies on the use of mathematical and/or computational models

Freeman stated that because of this generality. Social Network Analysis cuts across the boundaries of traditional disciplines. It therefore brings together sociologists, anthropologists, mathematicians, economists, political scientists, psychologists, communication scientists, statisticians, ethologists, epidemiologists and computer scientists.

2.3.2 History

Although there has been some mention of certain concepts of Social Network Analysis since the ancient Greeks, its main development has been done by two independently working research units in the 1930s.

Jacob Levy Moreno, an Austrian scientist who emigrated to the United States, is deemed to be the founder of *sociometry*. This technique is known as the systematic recording, graphical representation and analysis of social interaction in groups. However, a main part of his published work was carried out and planned by his colleague Helen Jennings. In his study at the *Hudson School for Girls* in 1934 he used the term *network* in the sense that it is used today. His studies included all four of the defining properties of Social Network Analysis: they were based on structural intuitions, they involved the collection of systematic empirical data, graphic imagery was an integral part and they included an explicit mathematical model [15].

[1] http://orgnet.com/sna.html

Approximately at the same time, another research group at the University of Harvard led by *W. Lloyd Warner* and *Elton Mayo*, focused on the research of social structure.

2.4 Betweenness and Group Betweenness Centrality

Generally within networks there are several measurements to describe a network's characteristics. Examples thereof are density, closeness or centrality. The central concepts of this Master's Thesis are the two measurements *Betweenness Centrality* (BC) and *Group Betweenness Centrality* (GBC).

Definition 1. *Betweenness Centrality (BC) is a good approximation for the quantity of information passing through a vertex in a communication network. Furthermore it can be used to estimate the potential monitoring and control capabilities a vertex may have on data flowing in a network [26].*

The BC of a vertice v can be calculated by dividing the sum of all shortest paths that involve that vertice v by the number of all shortest paths within the entire network.

The betweenness centrality of node v therefore is

$$BC(v) = \sum_{s \neq t \neq v} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} \tag{2.1}$$

whereas s and t are two vertices in the graph and $\sigma_{s,t}$ is the total number of shortest paths between the vertices s and t. Of all shortest path between s and t, several may pass through v. These are denoted as $\sigma_{s,t}(v)$. A basic way to calculate BC is by enumerating all shortest paths. An example therefore can be found in section 2.5.2 (equations 2.8 to 2.11).

Group Betweenness Centrality (GBC) is an enhanced concept of Betweenness Centrality. It was first defined by *Everett* and *Borgatti* and is notated as GBC [12].

Definition 2. *Group Betweenness Centrality (GBC) is a natural extensions of Group Betweenness. It can estimate the influence of a group of vertices over the information flow in a network.*

The betweenness centrality of group M is

$$GBC(M) = \sum_{s \neq t} \frac{\sigma_{s,t}(C)}{\sigma_{s,t}} \tag{2.2}$$

whereas s and t are two vertices in the graph and $\sigma_{s,t}$ is the total number of shortest paths between the vertices s and t. Of all shortest path between s and t, several may pass through any vertice in the group C. These are denoted as $\sigma_{s,t}(M)$. Again the basic way to calculate BC is by enumerating all shortest paths. Let us assume $M = (a, b)$ of the example network presented in figure 2.2. The formula for calculating $BC(M)$ can be seen in equation 2.3. The additional multiplication by 2 is justified by the undirected nature of the network [28].

$$GBC(M) = \left(\frac{\sigma_{a,b}(M)}{\sigma_{a,b}} + \frac{\sigma_{a,c}(M)}{\sigma_{a,c}} + \frac{\sigma_{a,d}(M)}{\sigma_{a,d}} + \frac{\sigma_{a,e}(M)}{\sigma_{a,e}} + \frac{\sigma_{b,c}(M)}{\sigma_{b,c}} + \frac{\sigma_{b,d}(M)}{\sigma_{b,d}} + \frac{\sigma_{b,e}(M)}{\sigma_{b,e}} + \frac{\sigma_{c,d}(M)}{\sigma_{c,d}} + \frac{\sigma_{c,e}(M)}{\sigma_{c,e}} + \frac{\sigma_{d,e}(M)}{\sigma_{d,e}} \right) \times 2 \tag{2.3}$$

By enumerating all shortest paths of the network and inserting these values into the equation, the Group Betweenness Centrality is

$$GBC(M) = \left(\frac{1}{1} + \frac{2}{2} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{0}{1} + \frac{2}{2} + \frac{1}{1} \right) \times 2 \tag{2.4}$$

which results in

$$GBC(M) = 9 \times 2 = 18 \tag{2.5}$$

As can be seen from such a small network, the counting of shortest paths can be an extensive process. It needs to be considered that the complexity of this process gives rise to a multiple with the size of the group M.

2.5 Computation of Group Betweenness Centrality

In the paper "*Fast algorithm for successive computation of group betweenness centrality*" a new algorithm for calculating the GBC was proposed by *Puzis et al.* In comparison with other algorithms its running time after preprocessing does not depend on network size. This algorithm is used as a basic foundation for the calculation of betweenness values.

To understand the concept of the algorithm a few definitions and notations are necessary. The algorithm assumes an unweighed, undirected and connected graph, which is defined as subject network $G = (V, E)$. Capital Latin letters are used to indicate unordered sets of vertices (C, M, V). By contrast, small Latin letters are used to indicate vertices (v, x, y, z).

2.5.1 Algorithm in detail

The algorithm itself is separated into two parts - preprocessing and computation. During preprocessing, three matrices needed for later calculation are generated. This first step is described in **Part A - Preprocessing** in more detail.

The second part - the actual computation - follows the algorithm presented in Algorithm 1 below.

Algorithm 1 Rami Puzis et. al. calculating GBC

Input: Group C, Distances $d^{n\times n}$, shortest paths $\sigma^{n\times n}$ Path betweenness's $\tilde{B}^{n\times n}$
Output: $GBC(C)$

1: Temporary group $M \leftarrow 0$
2: $\sigma^M_{x,y} \leftarrow \sigma_{x,y}, \forall x, y \in C$
3: $\tilde{B}^M(x,y) \leftarrow \tilde{B}(x,y), \forall x, y \in C$
4: Result variable $GBC \leftarrow 0$
5: **for all** $v \in C$ **do**
6: $\quad GBC \leftarrow GBC + \tilde{B}^M(v,v)$
7: $\quad$ **for all** $x, y \in C$ **do**
8: $\quad\quad \sigma^{M\cup(v)}_{x,y} \leftarrow \sigma^M_{x,y} - \sigma^M_{x,y}(v)$
9: $\quad\quad$ **if** $x = y \neq v$ **then**
10: $\quad\quad\quad \tilde{B}^{M\cup(v)}(x,x) \leftarrow \tilde{B}^M(x,x) - \tilde{B}^M(v,x) - \tilde{B}^M(x,v)$
11: $\quad\quad$ **else**
12: $\quad\quad\quad \tilde{B}^{M\cup(v)}(x,y) \leftarrow \tilde{B}^M(x,y) - \tilde{B}^M[(x,y) \circ v]$
13: $\quad\quad$ **end if**
14: $\quad$ **end for**
15: $\quad M \leftarrow M \cup (v)$
16: **end for**

Input: In addition to the three matrices d, σ and $\tilde{B}$, which, as mentioned before, are generated in the preprocessing, a network represented as a group of vertices C is essential as input.

Step 1 - Initialisation: Lines 1-4 of the algorithm initialise the variables M, σ^M, $\tilde{B}^M$ and the result variable GBC. σ^M and $\tilde{B}^M$ are simply copies of the shortest path $\sigma^{n\times n}$ and path betweenness $\tilde{B}^{n\times n}$ matrices from the input. The temporary group M is always a subset of C and contains all vertices, which were already accounted for their GBC contribution. M grows k times (size of the group C) until it is equal to C.

Step 2 - Add GBC contribution: The computation of the GBC value starts in line 5 with the inception of a loop. This loop roams through all vertices in group C. The first statement of the loop in line 6 adds the contribution of the vertices to the result variable GBC. This value is taken from the $\tilde{B}^M$ matrix.

Step 3 - Update σ^M: To ensure the correct update of $\tilde{B}^M$ the matrix σ^M first needs to be updated. This process starts with a loop in line 7 and is carried out in line 8. The update is done by reducing all values of the σ^M matrix by the shortest path passing through the vertice in procession.

Step 4 - Update $\tilde{B}^M$: Analogous to the update of σ^M, $\tilde{B}^M$ also needs to be recalculated. If the vertice pair from the matrix is in the form of $(x - x)$, the new value is simply a subtraction of the old value and the value of the vertice pair $(x - v) \times 2$.
In any other case the old value is reduced by the new path betweenness value $\tilde{B}$ of the constructed set between (x, y) and v. The construction of such an ordered group of three vertices follows the rule in equation 2.6.

$$(x,y) \circ z = \begin{cases} (z,x,y) & \text{if: } d(z,y) = d(z,x) + d(x,y), \\ (x,z,y) & \text{if: } d(x,y) = d(x,z) + d(z,y), \\ (x,y,z) & \text{if : } d(x,z) = d(x,y) + d(y,z). \end{cases} \tag{2.6}$$

If two or more conditions are satisfied any one can be taken. If no condition is satisfied (there is no shortest path that traverses x, y and z) then $(x, y) \circ z$ is not defined and the path betweenness value $\tilde{B}[(x, y) \circ z]$ is 0.

In case one condition is satisfied, the value of $\tilde{B}^M$ value can be calculated as stated in formula 2.7.

$$\tilde{B}^M(a,b,c) = \frac{\sigma^M_{a,c}(b)}{\sigma^M_{a,c}} \times \tilde{B}^M(a,c) \tag{2.7}$$

Step 5 - Grow M Line 15 adds the now fully processed vector v to the temporary group M.

As a result, this algorithm produces the GBC value of the total group C. In this work the algorithm is only partly used for calculating the GBC value of a group.

2.5.2 Example

For a better understanding of the algorithm, a simple example is fully calculated and executed.

Let us assume the example network is equal to the one presented in figure 2.2. The whole graph of this input network is referred to as group C.

As mentioned earlier, in order to be able to execute the algorithm, preprocessing of three matrices is necessary. This is designated as Part A below. Afterwards the actual computation, designated as Part B, is carried out.

Part A - Preprocessing

The following three matrices are calculated in this step:

- d: An $n \times n$ matrix, whose elements $d(s,t)$ store the distance between vertices s and t.
- σ: An $n \times n$ matrix, whose elements $\sigma_{s,t}$ store the number of available shortest paths between vertices s and t.
- $\tilde{B}$: An $n \times n$ matrix, whose elements $\tilde{B}(x,y)$ store the path betweenness of pair (x,y).

Figure 2.5 shows the matrices d and σ. These two matrices are simply received by algebraic path counting (viz. counting edges and shortest paths). As an example the edge between vertices d and e is observed. As can be seen in the example graph (figure 2.2) the shortest way from d to e is over two edges. Therefore the value of d - the distance - is 2. However, the information as to how many of such shortest path with the length of two exist is stored in the matrix σ. In this example there is only one path with distance 2: starting from d, going to a and finishing in e.

d	a	b	c	d	e
a	0	1	2	1	1
b	1	0	1	1	2
c	2	1	0	1	3
d	1	1	1	0	2
e	1	2	3	2	0

σ	a	b	c	d	e
a	1	1	2	1	1
b	1	1	1	1	1
c	2	1	1	1	2
d	1	1	1	1	1
e	1	1	2	1	1

Figure 2.5: Matrices d and σ of example network N

The matrix $\tilde{B}$ takes more effort in calculation. To be able to simplify this process the following two statements are taken as given [27]:

1. $\tilde{B}(x,y) = \tilde{B}(y,x)$
2. $\tilde{B}(x,x) = \mathrm{BC}(x) = \mathrm{BC}(y)$

The first statement reflects the undirected nature of the example network. An entry of the matrix does not need to be calculated twice for each direction (viz. value for $d - e$ and $e - d$ is equal). The second assumption gives a relief in calculation. To determine the path betweenness $\tilde{B}$ value for pairs like (a, a) or (e, e) instead of the

usual complex process, only the simple to obtain Betweenness Centrality needs to be calculated.

As an example the path betweenness $\tilde{B}$ of pair (a, a), which, in this case, is equal to the BC of vertice c is calculated in the following paragraphs. BC is defined by dividing the sum of all shortest paths that involve vertice v by the number of all shortest paths within the entire network. The mathematical notation is presented in equation 2.8.

$$BC(v) = \sum_{s \neq t \neq v} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} \tag{2.8}$$

In this concrete example for vertice a, the formula for calculating the BC is shown in equation 2.9. The additional multiplication by 2 is justified by the undirected nature of the network [28].

$$\begin{aligned} BC(a) = \Bigg(& \frac{\sigma_{a,b}(a)}{\sigma_{a,b}} + \frac{\sigma_{a,c}(a)}{\sigma_{a,c}} + \frac{\sigma_{a,d}(a)}{\sigma_{a,d}} + \frac{\sigma_{a,e}(a)}{\sigma_{a,e}} + \frac{\sigma_{b,c}(a)}{\sigma_{b,c}} + \\ & \frac{\sigma_{b,d}(a)}{\sigma_{b,d}} + \frac{\sigma_{b,e}(a)}{\sigma_{b,e}} + \frac{\sigma_{c,d}(a)}{\sigma_{c,d}} + \frac{\sigma_{c,e}(a)}{\sigma_{c,e}} + \frac{\sigma_{d,e}(a)}{\sigma_{d,e}} \Bigg) \times 2 \end{aligned} \tag{2.9}$$

By inserting the values from matrix $\sigma^{n \times n}$ into equation 2.9, the result is calculated by satisfying equation 2.10.

$$BC(a) = \left(\frac{1}{1} + \frac{2}{2} + \frac{1}{1} + \frac{1}{1} + \frac{0}{1} + \frac{0}{1} + \frac{1}{1} + \frac{0}{1} + \frac{2}{2} + \frac{1}{1} \right) \times 2 \tag{2.10}$$

which results in:

$$BC(a) = 7 \times 2 = 14 \tag{2.11}$$

This easy way of calculation can be carried out for all pairs like (a, a). For all other vertice combinations, the path betweenness $\tilde{B}$ cannot be determined by merely calculating the Betweenness Centrality. In such cases equation 2.12 is used to obtain the path betweenness value $\tilde{B}$.

$$\tilde{B}(x, y) = \sum_{s \in V} \delta_{s.}(y) \frac{\sigma_{s,y}(x)}{\sigma_{s,y}} \tag{2.12}$$

whereas

$$\delta_{s.}(y) = \sum_{t \in V | s \neq t} \frac{\sigma_{s,t}(y)}{\sigma_{s,t}} \tag{2.13}$$

As an example, the vertice pair (c, e) is used. Therefore the general equation 2.12 will be adjusted to equation 2.14.

$$\tilde{B}(c, e) = \sum_{s \in V} \delta_{s.}(e) \frac{\sigma_{s,e}(c)}{\sigma_{s,e}} \tag{2.14}$$

How to resolve the fraction part of the equations was already shown above. To resolve $\delta_{s.}(e)$ every possible s needs to be calculated separately. As shown below, variable s can reach the values (a, b, c, d, e).

$s = a$	$s = b$	$s = c$
$\delta_{a.}(e)\frac{\sigma_{a,e}(c)}{\sigma_{a,e}} = \delta_{a.}(e)\frac{0}{1} = 0$	$\delta_{b.}(e)\frac{\sigma_{b,e}(c)}{\sigma_{a,e}} = \delta_{b.}(e)\frac{0}{1} = 0$	$\delta_{c.}(e)\frac{\sigma_{c,e}(c)}{\sigma_{c,e}} = \delta_{c.}(e)\frac{2}{2} = ?$
$s = d$	$s = e$	
$\delta_{d.}(e)\frac{\sigma_{d,e}(c)}{\sigma_{d,e}} = \delta_{d.}(e)\frac{0}{1} = 0$	$\delta_{e.}(e)\frac{\sigma_{e,e}(c)}{\sigma_{e,e}} = \delta_{e.}(e)\frac{0}{1} = 0$	

The solution $\delta_{c.}(e)$ needs further computation, because there is no zero value in the numerator. It can be resolved by using equation 2.15.

$$\delta_{c.}(e) = \sum_{t \in V | c \neq t} \frac{\sigma_{c,t}(e)}{\sigma_{c,t}} \tag{2.15}$$

Again here an insertion for the Σ variable t is needed (as above for s). Variable t can take on the values (a, b, d, e) but not c.

$t = a$	$t = b$	$t = d$	$t = e$
$\frac{\sigma_{c,a}(e)}{\sigma_{c,a}} = \frac{0}{2} = 0$	$\frac{\sigma_{c,b}(e)}{\sigma_{c,b}} = \frac{0}{1} = 0$	$\frac{\sigma_{c,d}(e)}{\sigma_{c,d}} = \frac{0}{1} = 0$	$\frac{\sigma_{c,e}(e)}{\sigma_{c,e}} = \frac{2}{2} = 1$

The value of $\delta_{c.}(e)$ is therefore 1. When inserting this the total result is shown in equation 2.16.

$$\delta_{c.}(e)\frac{\sigma_{c,e}(c)}{\sigma_{c,e}} = \delta_{c.}(e)\frac{2}{2} = 1 \times \frac{2}{2} = 1 \tag{2.16}$$

The last step is to accumulate all calculated values together. As they are all 0 except the last value, the path betweenness $\tilde{B}(c, e)$ is 1. This is shown in equation 2.17.

$$\tilde{B}(c, e) = 0 + 0 + 1 + 0 + 0 = 1 \tag{2.17}$$

The fully calculated $\tilde{B}$ table is shown in figure 2.6. Depending on the problem, there may be no need to update the whole table for every progression step [27].

B~	a	b	c	d	e
a	14	3	2	3	4
b	3	8	2	1	2
c	2	2	8	2	1
d	3	1	2	10	2
e	4	2	1	2	8

Figure 2.6: Fully calculated initial matrix $\tilde{B}^{n \times n}$

Part B - Computation

The second part of the algorithm is dedicated to the actual calculation of the desired GBC value. For this part the steps from section 2.5.1 are processed sequentially.

Part B: Step 1 - Initialisation

The initialisation process is exactly as described in section 2.5.1.

Part B: Step 2 - Add GBC contribution

The loop starting in line 5 of the algorithm is continued by the addition of the GBC contribution to the result variable. This contribution is taken every loop round from the current (vic. updated) $\tilde{B}^{n\times n}$ matrix.

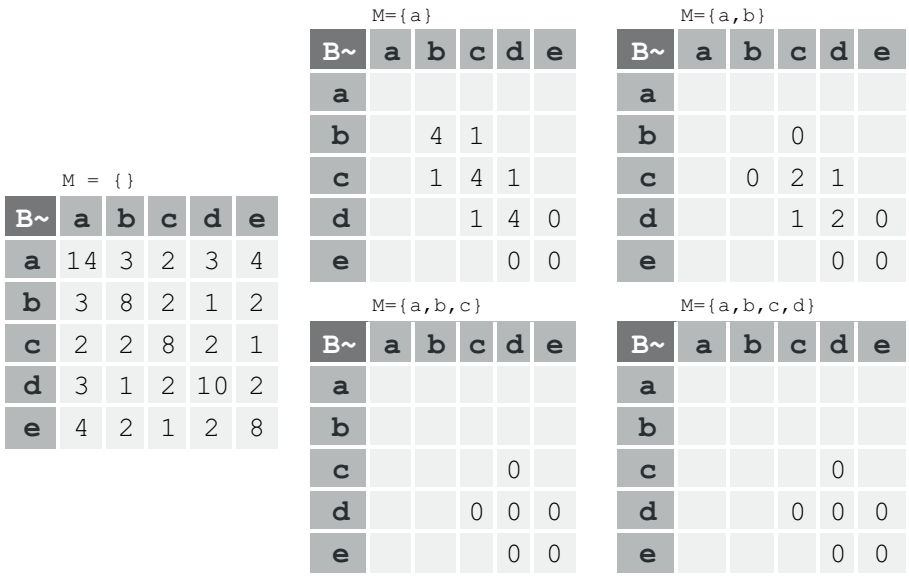

Figure 2.7: Updated $\tilde{B}^{n\times n}$ matrices

Let us assume the order in which all vertices of group C are added to group M is a, b, c, d and lastly e. In the first round the path betweenness value for a is added to the result variable GBC. The value for this step can be found by looking up row a and column a of matrix $\tilde{B}$ as shown in figure 2.7. The value is 14. This is carried on until the last vertice e is added. Equation 2.18 shows this operation until the loop is finished. It represents the GBC value of the group $C = (a, b, c, d, e)$.

$$GBC(C) = 14 + 4 + 2 + 0 + 0 \tag{2.18}$$

As can be seen in figure 2.7, none of the matrices is fully calculated The other values are simply not needed for the progression order a, b, c, d, e and therefore no effort is made to do so.

Part B: Step 3 - Update Sigma

To be able to carry out the updates of $\tilde{B}$ in the next step, the σ tables first must be updated (line 8 of the algorithm). This is done by simply recounting the available paths but without the recently added vertice. After each round, the matrix $\sigma^{n\times n}$ will look as shown in figure 2.8.

M = {}

σ	a	b	c	d	e
a	1	1	2	1	1
b	1	1	1	1	1
c	2	1	1	1	2
d	1	1	1	1	1
e	1	1	2	1	1

M={a}

σ	a	b	c	d	e
a	0	0	0	0	0
b	0	1	1	1	0
c	0	1	1	1	0
d	0	1	1	1	0
e	0	0	0	0	1

M={a,b}

σ	a	b	c	d	e
a	0	0	0	0	0
b	0	0	0	0	0
c	0	0	1	1	0
d	0	0	1	1	0
e	0	0	0	0	1

M={a,b,c}

σ	a	b	c	d	e
a	0	0	0	0	0
b	0	0	0	0	0
c	0	0	0	0	0
d	0	0	0	1	0
e	0	0	0	0	1

M={a,b,c,d}

σ	a	b	c	d	e
a	0	0	0	0	0
b	0	0	0	0	0
c	0	0	0	0	0
d	0	0	0	0	0
e	0	0	0	0	1

Figure 2.8: Updated $\sigma^{n \times n}$ matrices

Part B: Step 4 - Update B Tilde

Lines 9 to 13 represent the update of $\tilde{B}^{n \times n}$ matrix. This process is carried out exactly as described in the initialisation phase described in **Part A**.

Part B: Step 5 - Growing MG

Following all updated, group M is increased by the last vertice treated. Equation 2.19 shows this process.

$$\begin{aligned} M &= \{\} \\ M &= \{a\} \\ M &= \{a, b\} \\ M &= \{a, b, c\} \\ M &= \{a, b, c, d\} \\ M &= \{a, b, c, d, e\} \end{aligned} \tag{2.19}$$

Final result

After the last vertice and its contribution was added to the associated variables, the algorithm finishes. Equation 2.18 shows the GBC value of group C (which in this case represents the whole network of figure 2.2) is 20.

3 Datasets

To put an algorithm into practice, test data is needed. This data can either be obtained by synthetic generation or abstracted from real world models.

3.1 Real world data

Real world data can have its origins in any social observation object that has any kinds of ties to other objects of the same type. In the age of computers the challenge is not the collection of the data itself; scripts can easily be written to execute that task. The more complex parts are the choice of the subject of study, the parameters and the boundaries of the data. First of all, a subject of study must be chosen, for example the faculty staff of a university department. At first glance the boundaries of such a set seem obvious: every faculty member at the faculty should represent an actor. However, an exact description of "employed" must first be found. Should only full-time employees be considered or also part-timers? What about teaching assistants and Ph.D. candidates? Wassermann also states that in addition there is the problem that actors may come and go and the total number may be difficult to enumerate" [35]. These steps rely heavily on the expertise and knowledge of the researcher and must be carefully considered.

3.1.1 BuddyPress

The first data set which is used in this thesis for testing purposes was derived from a social network in the classical sense. Technically the social network is based on a *WordPress* plugin called *BuddyPress*[1]. This plugin enhances the blog system *WordPress* with social network functionality, like friendships, private messages and profile pages. The data set is derived from the social network called *Old Boys Alliance*[2]. Its main purpose is to support networking and communication between the members of an Austrian eSports club.

The data set itself consists of 92 actors and those comprise 405 relations. Initially the subject of study also included two actors which were described as inactive by

[1] http://buddypress.org/
[2] www.oldboysalliance.at

other members of the network and did not have any relations to other actors. Those two vertices were not included in the the final data set. Figure 3.1 shows the data set in the graph theoretic representation form.

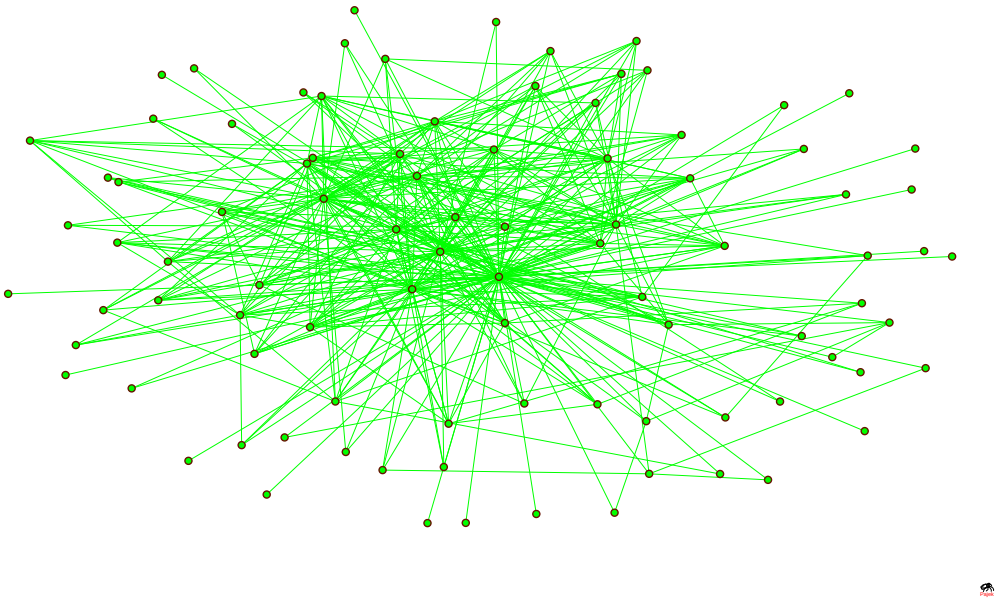

Figure 3.1: BuddyPress data set in graph notation

3.1.2 US airports

The second real world data set is based on the US-American domestic air traffic of 2006. It consists of the 500 busiest commercial airports (vertices) and their 5960 flight connections (edges)[3].

Originally the network was directed. However, as *Barrat* et al. already stated, even though this type of network is directed by nature as a flight is scheduled from one airport to another, the network is highly symmetric [2]. Therefore this version of the network is nondirectional.

Also initially the network was weighted, and those weights corresponded to the number of seats available on the scheduled flights, though the proposed algorithm only supports dichotomous (unweighed) networks. Therefore the network was consolidated and all edges were transformed into dichotomous connections. There have been considerations to introduce a boundary for the transformation into a dichotomous network when a connection is existent or not. A boundary itself (median or average for example) would not have changed the general structure and characteristics of the network. However, without a boundary (or a boundary set to 0) the

[3]Source: sites.google.com/site/cxnets/usairtransportationnetwork

network offers better quality for testing, simply because of its size. Any introduction of a boundary would have reduced its complexity and sheer size.

3.2 Synthetically generated data

In order to not be concerned with the issues described above regarding the sampling of real world data, synthetically generated data is used during testing periods. As within this thesis, the tool *Pajek* and its native data format *.net* is already used for representation and data exchange, its now also applied for generating random graphs (see: [4]).

The study of random networks has long been dominated by the *Erdős-Rényi* model $G(n, p)$, where an edge between a pair of vertices n is present with a certain probability p [6]. It is an assumption that such a mathematical model can never recreate the nature and characteristics of real world social networks. Therefore also the more complex model by Albert and Barabási for creating random graphs is consulted [1].

Another interesting network type in connection with social network analysis due to it's characteristics are *Small World* networks. In such networks most nodes are not direct neighbours, but most of them can be reached by every other by a defined small number of steps. *Pajek* uses the approach by *Brandes* and *Batagelj* to create such networks [3].

The following additional synthetic test data sets were created:

- **High Density Network:** Based on the *Erdős-Rényi* model this data set contains 200 vertices with every of them beeing connected to a minimum of 75 and a maximum of 125 edges. In total 19,753 edges were created.
- **Small World Network:** This network consists of 250 edges and their corresponding 500 edges. The inital number of edges on each side of a vertice 2 is altered by the replacement probablity of 0.5 accordingly to the previously mentioned work by *Brandes* and *Batagelj* [3].
- **Extended Model Network:** This network is based on the extended model of *Albert* and *Barabási*, It contains 200 vertices, 794 edges and uses 2 isolated nodes, a probability of 0.3 to add new edges and 2 lines to be added at each steps as parameters for creation [1].

4 Algorithm

4.1 Complete Enumeration

The complete enumeration algorithm is a simple and well-known technique. The method sets up all possible combinations and calculates the fitness value for each solution. Due to this completeness it can thus guarantee finding the optimum solution. On the other side, for np-hard problems this approach reaches its limits quickly. The runtime to find a solution can easily move into unacceptable ranges.

Algorithm 2 shows the implemented complete algorithm as a pseudocode. The aim of the algorithm is to find a group C of size k (number of vertices group C consists of) with the highest possible GBC value. This group C needs to be a subset of network N. Also more than one group may have the highest GBC value in the network. The two methods *checkLinkedGroupConstraint* and *gbcRequest* refer to shared library methods for all implemented algorithms. The first one uses Algorithm 1 to calculate GBC values. The second method validates to which extent a given group (set of vertices) fulfills the clique constraint.

Algorithm 2 Complete Enumeration to find a clique with high GBC

Input: Network N, size of group C denoted as k
Output: Highest possible GBC value GBC

```
1:  GBC of current combination: currGBC ← −1
2:  Highest GBC found: xBestGBC ← −1
3:  Groups with high GBC: xBestCombination ← 0
4:  for all combinations C ∈ N with size k do
5:      if checkLinkedGroupConstraint(C) == true then
6:          currGBC ← gbcRequest(C)
7:          if currGBC == xBestGBC then
8:              xBestCombination ∪ C
9:          end if
10:         if currGBC > xBestGBC then
11:             xBestGBC ← currGBC
12:             xBestCombination ← C
13:         end if
14:     end if
15: end for
```

For all test data networks the complete enumeration has been executed for each size of k. The results are shown in table 4.1, whereas the data set named *Extended Model* (EXT), *High Density* (HD) and *Small World* (SW) are synthetically generated. All other networks are based on real world data (see chapter 3). Depending on the network size the runtime of the complete algorithm begins reaching unacceptable dimension already by $k = 5$.

Dataset	k	Runtime ms	Fitnessvalue (GBC)	Result Vertices
BuddyPress	2	54.05	6,384.75	[12,13]
	3	91.81	6,605.48	[12,13,19]
	4	1,097.42	6,787.09	[6,12,13,19]
	5	27,768.87	6,909.53	[6,11,12,13,19]
	6	6,158,139.91	7001.68	[5,6,11,12,13,19]
	7	*Stopped after 6 hours 10 minutes (22,200,000 ms)*		
Extended Model	2	79.19	24,631.71	[1,3]
	3	366.03	31,411.06	[1,3,7]
	4	23,481.54	36,847.67	[1,3,7,8]
	5	*Stopped after 4 hours 34 minutes (16,440,000 ms)*		
High Density	2	584.55	923.11	[20,87]
	3	2,877.86	1,377.92	[20,87,140]
	4	148,113.24	1829.20	[20,87,131,140]
	5	*Stopped after 14 hours 5 minutes (50,700,000 ms)*		
Small World	2	96.30	6,225.38	[12,233]
	3	502.58	5,780.58	[155,156,156]
	4	*No clique available*		
Airport	2	2,589.46	84,534.87	[7,56]
	3	60,335.97	112,320.80	[7,14,56]
	4	13,102,339.04	136,912.79	[6,7,14,56]
	5	*Stopped after 14 hours 51 minutes (53,460,000 ms)*		

Table 4.1: Complete enumeration runtimes for various k values

4.2 Simulated Annealing

In the last section it was shown that a complete enumeration of all solutions is a time-consuming process. Furthermore, depending on the size of the network and on the size of k, the calculation time in many cases is disproportionate to the result or it is even not possible to solve the problem in a finite time span. Therefore a meta heuristic approach - a method which does not guarantee to find the optimum solution, but instead drastically reduces computation time - is favoured. Such a method tries to improve an initial solution step by step with regard to a given measure of quality (also called fitness or energy level). Despite the fact tat it may be impossible to find the optimum solution to the problem, a meta heuristic approach can deliver an acceptable approximation in a fixed amount of time.

Simulated Annealing (SA) is such a meta heuristic approach for solving optimization problems. This probabilistic method was proposed independently by *Kirkpatrick*, *Vecchi* and *Gelatt* in 1983 [21] and by *Černy* in 1985 [34]. The method is inspired by the metallurgy annealing process, a technique where a metal is heated and then cooled down in a controlled manner. The slow cooling process gives the crystals of the solid sufficient time to rearrange themselves into a new structure with an overall lower internal energy than in the previous state. After the cooling process is finished, the structure is in a state of minimal energy which should be close to the optimum.

Simulated Annealing follows the basic procedure described in algorithm 3. To use this technique, an initial solution (x) has to be defined, in which x is always a subset of the given network (graph). With x as input, the loop in line 2 is repeated until a termination condition. This condition can either be a specific amount of time, the achievement of a target value or reaching a certain final temperature. If the cost of the new neighbour solution (x') decreases (or increases when trying to find a global maximum), then the solution is changed and the move is accepted (line 5). Otherwise, the move is accepted only with a probability depending on the cost decrease and a control parameter called temperature t (line 7) [5].

Algorithm 3 General scheme of Simulated Annealing

```
1:  x ← x(0)
2:  while termination condition not reached do
3:      t ← t'
4:      x' ← neighbor(x)
5:      if (if(x') > f(x)) then
6:          x ← x'
7:      else(P(f(x), f(x'), t) > random())
8:          x ← x'
9:      end if
10: end while
```

The probability function to accept moves must provide that when t tends to zero, the probability $P(f(x), f(x'), t)$ must also tend towards zero. Additionally the probability is high at the beginning to allow the algorithm to escape from local minima. This so called acceptance probability was defined by Kirkpatrick in [21] as follows:

$$P(f(x), f(x'), t) = \begin{cases} 1 & \text{if: } f(x') < f(x), \\ e^{\frac{e-e*}{t}} & \text{otherwise.} \end{cases} \tag{4.1}$$

Simulated Annealing is a proven and prominent technique. Therefore this method has been chosen to be applied to the MGBC problem

4.2.1 Algorithm design

The algorithm follows the general simulated annealing schema presented in algorithm 3. It accepts as parameters the size of the group (k), an input network in *Pajek* format *.net*, and variables regarding the execution of the algorithm. In the following sections the specific design elements of the algorithm are discussed in detail.

Choice of Neighbourhood

The choice of a neighbourhood structure has a great influence on the behaviour and runtime of the SA algorithm. In the case of the MGBC problem, networks are too complex and small changes may have unpredictable consequences. Therefore no specific restrictions can be made regarding interchanging of specific members. Thus the the choice of a neighbour simply works by choosing a random member and exchanging it against another random member, which is not yet present in the solution.

Initial temperature

The choice of an initial temperature is based on the work of *Thompson* and *Bilbro* [33]. To find the initial temperature T_0, the following probability distribution was used:

$$p = e^{\frac{\Delta E}{T_0}} \tag{4.2}$$

where ΔE is the average fitness of ten random solutions plus their standard deviation. The probability p was set to 0.8. For one of the test networks used the distribution is presented in figure 4.1. Subsequently the initial temperature T_0 for

this case would be around 750. Additionally T_0 for k can never be lower than T_0 for any other smaller k value.

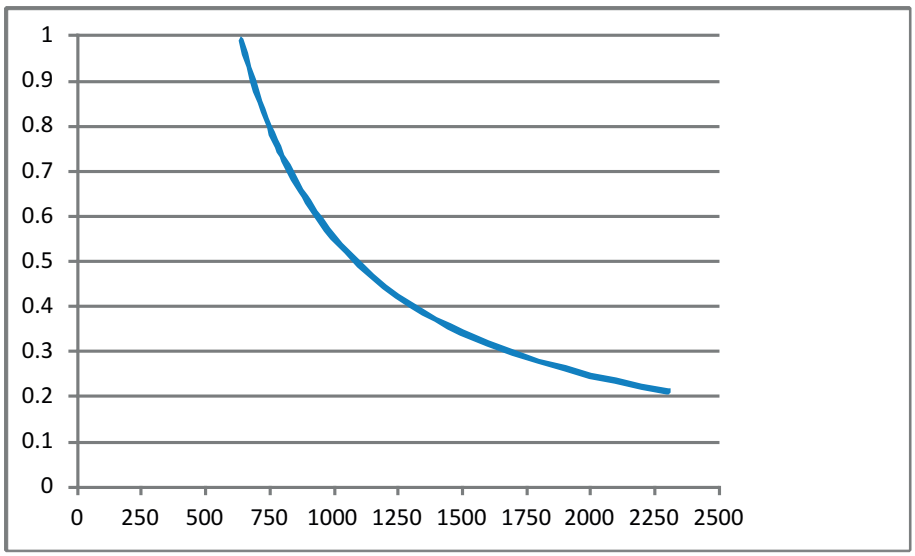

Figure 4.1: Probability distribution for T_0

Cooling Schedule

A practical Simulated Annealing implementation requires the generation of a finite sequence of values to decrease the temperature T_0, and a finite number L of state transitions. To achieve this aim a cooling schedule must be specified [24]. Such schedules are grouped into static schedules, which must be specified before the algorithm even begins and adaptive schedules, which adjust the temperature decrease depending on information gathered during the algorithm's execution [17].

Regarding the definition of a cooling schedules [30] states that the rate of decay should be low enough to give them enough time to form a crystal lattice with minimum internal energy (global optimum). However, slow cooling rates will automatically lead to a long runtime. On the other side, it is also not advisable to chose a temperature too low, as atoms of the molten metal would not have enough freedom to rearrange their positions and therefore no or nearly no global optimum will be found. *Shojaee*, *Behnam* and *Shakouri* also state that although there are some theoretical limits and formulations to choose a proper cooling rate, there is no deterministic criterion in the literature [30].

Strenski and *Kirkpatrick* suggest that optimal cooling schedules are not monotone in decreasing the temperature. They also show that geometric and linear cooling schedules perform better than inverse logarithmic cooling schedules when sufficient computing effort is allowed [32].

Based on this analysis, a temperature decrease function in the form of $t_* = t_0 * f$ is used. Furthermore on *Díaz* and *Sierra's* suggestion in [24] that f should generally be between 0.8 and 0.99, f was set to 0.95 for all approaches .

Termination Criteria

A cooling schedule also includes a finite number of state transitions L, hence the point at which the algorithm stops. To determine the number of steps the self-defined following function was used:

$$L = e^{k-1} * k^2 * t \tag{4.3}$$

where the step count is rounded to the thousandth digit and the maximum runtime may not exceed 10 minutes. As an example L for the network BuddyPress is presented in table 4.2.

Data set	k	e^{k-1}	$e^{k-1} * k^2$	t	L
BuddyPress	2	2.7182818285	10.8731273138	750	8.000
	3	7.3890560989	66.5015048904	1100	73.000
	4	20.0855369232	321.368590771	1570	504.000
	5	54.5981500331	1364.9537508286	1960	2.675.000
	6	148.4131591026	5342.8737276928	2550	13.624.000
	7	403.4287934927	19768.010881144	2550	50.408.000

Table 4.2: Calculation of state transitions L

4.2.2 Finding a feasible solution

The presented SA algorithm to solve the MGBC problem also comes across solutions which do not compose a clique. Such solutions are called infeasible and are not desired. To find a feasible solution, a suitable approach needs to be applied. Such an approach can be either to eliminate unacceptable solutions (*penalty function method*) or only generate feasible solutions (*repair function method*). As the generation of a feasible solution can be a hard task for this problem statement, the penalty function method was chosen for all algorithms.

A penalty method is implemented by reducing the fitness value of a solution by a specific penalty method value. This value is zero if the constraint is not violated at all. In all other cases, the value is greater than zero and can for example be

depending on runtime, previous results, k and/or severity of the violation. All implemented penalty methods have in common that the severity of the violation is a discrete metric.

In the following paragraphs the two applied penalty techniques are discussed in more detail.

Dynamic Penalty

A dynamic penalty function consists primarily of two aspects:

- A distance which increases the penalty, depending on the time already spent on the search progress.
- A dynamic aspect which increases the penalty, depending on the the severity of the constraint violation.

As *Schwarzer* and *Coit*, stated in [31] this method offers the advantage that highly infeasible solutions are also allowed in the early steps of the algorithm while continually increasing the penalty imposed to eventually move the final solution to the feasible region. This technique is also similar to the noising method presented by *Charon* and *Hudrey*, where the penalty function is described as noise and is reduced at each outer loop iteration of the algorithm [17].

The dynamic penalty method used follows equation 4.5, whereby the term in the first square brackets represents the distance and the second measures the severity of the violation.

$$p(f) = [(c \times \text{annealingSteps})^{\alpha}] \times \left[e^{\Omega - 1} \times \left(\frac{\text{fitness of best solution}}{k} \right) \right] \tag{4.4}$$

wherby

$$\Omega = \left[\left(\frac{\text{number of edges in solution}}{\text{total number of edges to fullfil clique constraint}} \right) - 1 \right]^2 \tag{4.5}$$

As recommend by *Joines* and *Houck* [20], the variables c and α are set to 0.5 respectively 1.

Hybrid Static-Dynamic Penalty

Static penalty methods apply a constant penalty to all solutions which violate the constraint in any way or severity. This approach uses this idea, but enhances it with a dynamic aspect to penalise the severity of the violation. In comparison with the dynamic penalty, the distance of the algorithm is not taken into account.

The applied penalty method follows equation 4.6.

$$p(f) = \left(e^{\Omega-1} \times \text{fitness of best solution}\right)^2 \tag{4.6}$$

The additional exponentiation is necessary to create a penalty value which is large enough to be recognisable. It is possible that this exponentiation is not required with certain networks.

4.3 Genetic Algorithm

The term *Genetic Algorithm* (GA) was first used by *John Holland* in his book "*Adaptation in Natural and Artificial Systems*" in 1975 [18]. Since then, genetic algorithms were applied to a wide range of problem scenarios and their popularity is still high among all disciplines.

Genetic Algorithms imitate the evolutionary behaviour of biological systems. They generate a sequence of populations containing possible solution candidates to the underlying optimisation problem [17]. By using transition operators like crossover or mutation, each generation is transformed into a descendant population.

Excursus: Biological Terminology

As Genetic Algorithms are derived from biology, the terminology of this field was also adopted. Therefore it is useful to introduce some of these terms, which will be later used within this thesis.

All living organisms consist of cells and each of this cells contain the same set of one or more *chromosomes*, which serve as a blueprint for the organism. A chromosome is divided into *genes* and each gene encodes a particular protein. Greatly simplified, one gene can be seen as a single trait of the organism such as eye colour. The possible values such a trait can take - e.g. blue, green, brown - are called *alleles*. Every allele is bound to a specific position within the gene called a *locus*. Many organisms have multiple chromosomes in each cell. The complete collection of chromosomes is called the organism's *genome* [25].

In the context of Genetic Algorithms, a chromosome refers a candidate solutions, which can be encoded as a bit string or with real values. Every single gene represents one number of the candidate solution. The alleles are the possible values a gene can take. In case of binary encoding the alleles are 0 and 1.

Crossover usually consists of exchanging genetic material between two parent candidate solutions. Mutation is carried out by flipping a bit or changing a value at a randomly chosen locus.

Algorithm 4 shows a generic template for a genetic algorithm. While selection and fitness evaluation must be executed for every generation, crossover and mutation depend on the defined criteria of the concrete implementation.

Algorithm 4 General scheme of a Genetic Algorithm

```
1: Choose initial population of chromosomes
2: while termination condition not reached do
3:     while until enough offspring is generated do
4:         if (Crossover condition satisfied) then
5:             Select parents
6:             Choose crossover parameters
7:             Perform crossover
8:         end if
9:         if (Mutation condition satisfied) then
10:            Choose mutation parameters
11:            Perform mutation
12:        end if
13:        Evaluate fitness of offspring
14:    end while
15:    Select new population
16: end while
```

4.3.1 Algorithm Design

The algorithm follows the general Genetic Algorithm scheme presented above. To execute the algorithm the input parameters k, a network N in *Pajek* format *.net* and variables regarding the execution of the algorithm are needed.

The parametrisation of a Genetic algorithm can reach incredible dimensions. Early on many researches used the recommended test suite (population of $50 - 100$, crossover rate of 0.6, mutation rate of 0.1% per bit) by De Jong and applied it to their problem without questioning if this suite could even fit [25].

hope that is a mathematical term, otherwise you would need something like incredible dimensions...

To track down this issue two approaches were selected to be carried out and later on to be compared. The first one - *generation approach* (GEN) - tries to keep the size of the population relatively low and unvaried. It runs over a long number of generations and therefore has a low crossover rate. The second approach - *population approach* (POP) - does the exact opposite. To solve the problem it uses a low and fairly constant number of generations with a high crossover rate, but increases the size of the population as k rises.

In the following sections the design elements of the algorithm are discussed in more detail.

Encoding

When designing a Genetic Algorithm, the first major decision is the choice of the encoding. It influences not only the runtime, but also determines how transition operators used later can be applied.

With *binary encoding* a solution candidate is represented as a string consisting of only zeros and ones. These values declare either a gene is present or not in the solution.

Another possibility to represent a chromosome is *real value encoding* (also sometimes called *permutation encoding*). There the genes are represented by the actual value. From this follows that the genes may be shorter, but the allele range is much larger.

Figure 4.2 shows the representation in both basic forms of the following example: The given network has 10 vertices $V = (1, 2, 3, 4, 5, 6, 7, 8, 9, 19)$, $k = 2$ and the solution to be represented is $[2, 10]$.

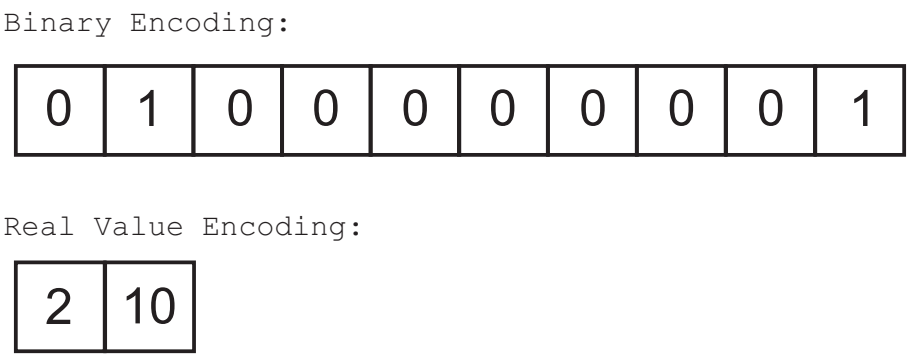

Figure 4.2: Example of binary and real value encoding

Most of the literature uses binary encoding, and many examples and papers are also based on binary encoding. Nevertheless the SA algorithm implemented here makes use of real value encoding.

The reason therefore is simple: in binary encoding the length of a chromosome is the size of the vertices present in the network which, in the case of the test data used will be between 92 and 500. With real value encoding the size of a chromosome is only equal to k, which ranges from 2 and 7.

Mitchell states in [25] that the performance of an encoding depends mainly on the problem itself and that there are currently no rigorous guidelines for predicting which encoding will work best.

Initial population

The choice of the initial population is random, meaning all chromosomes are generated with their genes without any presumptions or designation. This is due to the high complexity of the networks where predictions are impossible. If choosing an initial population for example based on the betweenness value of a vertice, the algorithm might stay within a local optimum and may not be able to succeed very well.

Selection

The selection of chromosomes for reproduction is a crucial step in the overall design of the algorithm. The original selection method *Roulette Wheel* proposed by *Holland* seems to be deprecated nowadays and no longer recommended any more [23]. Therefore it will be neither further considered nor explained.

Instead the following methods will be applied:

- **Elitism**: This concept was introduced by *De Yong*, a doctoral student of *Holland*, in [10]. A given elitism rate chooses the solutions with the best fitness value and simply clones them as new children. This method guarantees that good solutions already found are not discarded again.
- **Tournament**: Tournament selection chooses a solution candidate for reproduction within a tournament, whereas the number of competitors can be configured (initial value is 4). The competitor with the highest fitness value wins and is therefore selected.

The implemented algorithm uses an initial elitism rate of 0.2. All other new solution candidates are chosen by tournament selection.

Crossover

This operator represents the reproduction of two chromosomes mixing their genes to generate new offspring. A random locus is chosen and the sub-sequences before and after that locus assemble the new chromosome. The crossover can be performed with only one such described crossover point (denoted as $1X$) or with more (for example two-point crossover, denoted as $2X$).

The example in figure 4.3 shows a $1X$ crossover between two chromosomes. Two offspring are produced, though unfortunately *Offspring 2* is invalid because vertices can only occur once in a each solution chromosome. However, the value of locus 2 and 4 is 7 in both cases. Therefore additional mutation has to be carried out to solve this issue. The first occurring locus - in the example locus 2 - is chosen and its

value replace by a new random value which has not yet occured in the chromosome (4, 7 or 3 are no valid choices).

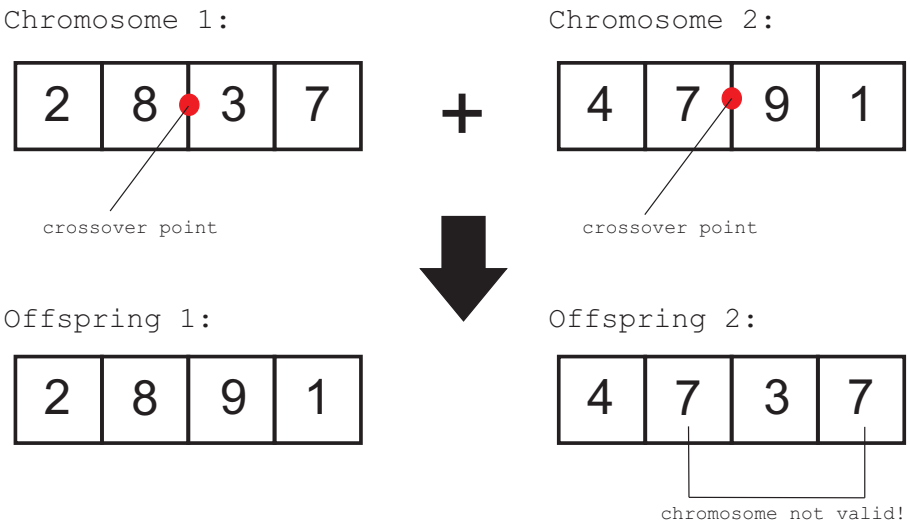

Figure 4.3: Crossover with a single crossover point at locus 2

Single crossover points are usually not recommend [11], [23]. Nevertheless this implementation uses such a single crossover point due to the small length of a chromosome, which ranges between 2 and 7. In the implementation this crossover point is not at a fixed locus (e.g. at the middle of the chromosome), instead it is determined independently and randomly for each crossover step.

All implementations use a *CrossoverORMutation* strategy. This means that crossover and mutation cannot happen in the same generation to the same chromosome. Excluded from this rule is the automatic mutation for receiving valid solutions described above.

For the generation approach the crossover rate χ is set to 0.2, for the population approach $\chi = 0.6$.

Mutation

This operator is responsible for random and unexpected changes in any chromosome to not only ensure diversity but also to perhaps find a better solution. As to its random characteristic and possible high impact, it should only occur very rarely.

In the proposed algorithm, the mutation rate is a probability which at each locus of each chromosome may exchange that gene with another random allele. This probability is usually quite low. Therefore the initial mutation rate μ for both approaches is set to 1%.

Termination Criteria

In principle, genetic algorithms could run forever. In practice however, a termination criteria is needed. Common approaches include to set a number of fitness evaluations or a specific amount of time.

The termination criteria for both approaches is simply the number of generations. Hence the challenge is to determine the right size of the population and the number of generations.

The size of the population has been approached from multiple theoretical points of view. *Reeves* states in [29] that the choice of a population size is always a trade-off between efficiency and effectiveness. Regardless of the problem, too small populations do not allow sufficient room for exploring the search space effectively. On the other hand, too large populations will impact the efficiency of the algorithm so that no solution can be found in a acceptable amount of time.

The size of the population and the number of generations for each approach can be seen in chapter 5.2

4.3.2 Finding a feasible solution

A Genetic Algorithm faces the same problems regarding feasible and infeasible solutions as Simulated Annealing. Therefore an elaborated penalty method is needed. *Yeniay* assembled all relevant penalty methods for Genetic Algorithms and discussed their strengths and weaknesses in [36]. Yeniay's work together with the experience gained from formulating penalty methods for Simulated Annealing resulted in two penalty functions. The following paragraphs discuss dynamic and the adaptive approaches in detail.

Dynamic Penalty

Analogous to the dynamic penalty function of the Simulated Annealing implementation, this penalty function also consists of two primary aspects:

- A distance which increases the penalty, depending on the time already spent on the search progress.
- A dynamic aspect which increases penalty, depending on the the severity of the constraint violation.

The dynamic penalty function used follows equation 4.7, whereby the term in the first square brackets represents the distance and the second measures the severity of the violation.

$$p(f) = [(c \times \text{generation steps})^{\alpha}] \times \left[e^{\Omega-1} \times \emptyset f(\text{last generation})\right] \tag{4.7}$$

wherby

$$\Omega = \frac{\text{number of edges in solution}}{\text{total number of edges to fullfil clique constraint}} \tag{4.8}$$

Again the variables c and α are as recommend by *Joines* and *Houck* [20] set to 0.5 respectively 1.

Adaptive Penalty

Adaptive penalty methods update their penalty parameter for every generation according to information gathered from the population [36].

The applied penalty method follows equation 4.9.

$$p(f) = \lambda * e^{\Omega-1} \tag{4.9}$$

whereas

$$\lambda(t+1) = \begin{cases} \left(\frac{1}{\beta_1}\right) \times \lambda(t) & \text{if: Case 1,} \\ \beta_2 \times \lambda(t) & \text{if: Case 2,} \\ \lambda(t) & \text{otherwise.} \end{cases} \tag{4.10}$$

Case 1 denotes that all of the best individuals (equal to elitism rate) in the last generation are feasible and *Case 2* denotes that they are infeasible. This leads to a reduction of the penalty term if all chromosomes of the last generation were feasible, but on the other hand to an increase if they were infeasible. *Hadj-Alouane* and *Bean* suggest $\beta_2 = 1.7$ and for β_1 either 2, 4 or 8 [16].

5 Results

This chapter compares the previously described meta heuristics internally between different penalty methods and externally against the other algorithm in quality and runtime.

5.1 Simulated Annealing

According to the proposed algorithm design in section 4.2.1, initial temperatures and annealing steps were calculated and tests carried out with two different penalty methods. An assembly can be found in the table below. Values marked with a star (*) were reduced to fit the criteria of a maximum runtime around ten minutes.

Simulated Annealing - T_0 and step numbers						
Data set	k	T_0	Steps	k	T_0	Steps
BuddyPress	2	750	8,000	5	1,960	2.675.000
	3	1,100	73.000	6	2,550	13,624,000
	4	1,570	504.000	7	2,550	12,602,000 (*)
Extended Model	2	1,375	14,000	5	4,720	644,2000
	3	2,400	159,000	6	4,720	14,834,000 (*)
	4	3,000	964,000	7	4,720 (*)	12,361,000 (*)
High Density	2	1,520	16,000	5	3,780	515,000
	3	2,280	151,000	6	3,780 (*)	7,200,000 (*)
	4	3,050	980,000	7	3,780 (*)	5,142,000 (*)
Small World	2	5,350	58,000	3	7,800	518,000
Airport	2	14,300	155,000	5	22,500	17,000,000 (*)
	3	14,300	950,000	6	22,500 (*)	15,500,000 (*)
	4	19,800	6,363,000	7	22,500 (*)	11,600,000 (*)

The results of the two penalty methods are presented in tables 5.1 and 5.2.

Simulated Annealing - Dynamic Penalty						
Data set	k	Highest	Lowest	Runtime ms	Deficit	% Deficit
BuddyPress	2	6,384.75	5,856.01	1,048.70	0.00	0.00
	3	6,605.48	6,504.17	2,621.86	0.00	0.00
	4	6,787.09	6,654.10	14,284.14	0.00	0.00
	5	6,909.532	6,740.10	86,875.52	0.00	0.00
	6	6,983.774	6,740.10	519,527.07	-	-
	7	0.00	0.00	633,722.00	-	-100.0.
Extended Model	2	24,631.71	19,241.82	1,593.66	0.00	0.00
	3	31,411.06	24,836.02	4,658.70	0.00	0.00
	4	31,640.76	26,167.90	25,686.30	5,206.91	-14.13
	5	36,915.67	26,062.26	20,1391.67	-	-
	6	0.00	0.00	595,249.15	-	-100.0
	7	0.00	0.00	585,748.46	-	-100.0
High Density	2	1,377.92	1,374.08	11,484.59	0.00	0.00
	3	1,826.36	1,823.67	54,662.08	0.70	-0.05
	4	2,267.68	2,261.48	39,742.99	63.85	-3.38
	5	2,267.68	2,261.48	40,031.92	-	-
	6	2,713.41	2,705.74	634,315.47	-	-
	7	3,145.89	3,135.49	590,034.63	-	-
Small World	2	6,225.39	6,225.39	2,507.65	0.00	0.00
	3	5,780.58	5,392.39	11,694.29	0.00	0.00
Airport	2	84,534.87	82,420.81	7,942.45	0.00	0.00
	3	109,933.15	76,680.82	27,585.88	2,387.65	-2.13
	4	102,867.62	79,936.17	182,236.76	34,045.17	-24.87
	5	109,700.71	63,633.72	605,296.30	-	-
	6	121,792.13	0.00	696,475.99	-	-
	7	0.00	0.00	696,475.99	-	-100.0

Table 5.1: Results of Simulated Annealing with dynamic penalty

The tables include the highest and lowest fitness as well as the average runtime in milliseconds. If an estimated global optimum is available the deficit in absolute and relative notation ins denoted. For each test case the algorithm was executed ten times.

Simulated Annealing - Hybrid Penalty						
Data set	k	Best	Weakest	Runtime ms	Deficit	% Deficit
BuddyPress	2	6,384.750	5,884.331	1,084.829	0.00	0.00
	3	6,605.476	6,576.335	2,544.230	0.00	0.00
	4	6,787.094	6,663.028	13,149.203	0.00	0.00
	5	6,838.801	6,721.147	85,079.828	70.73	-1.02
	6	6,953.242	6,834.575	509,263.475	-	-
	7	7,015.49	0.00	593,982.87	-	-
Extended Model	2	24,631.71	18,451.237	1,567.46	0.00	0.00
	3	31,411.06	24,832.96	4,599.21	0.00	0.00
	4	31,694.57	26,100.10	24,700.72	5,153.10	-13.98
	5	36,918.16	26,061.38	198,389.73	-	-
	6	0.00	0.00	605,052.21	-	-100.00
	7	0.00	0.00	600,757.31	-	-100.00
High Density	2	923.11	920.38	6205.78	0.00	0.00
	3	1,377.51	1374.16	11,583.07	0.29	-0.02
	4	1,825.58	1822.51	54,228.26	64,62	-3.42
	5	2,270.50	2263.03	43,179.56	-	-
	6	2,714.66	2,704.80	684,651.86	-	-
	7	3,147.89	3,138.01	599,851.86	-	-
Small World	2	6,225.39	6,225.39	2,575.21	0.00	0.00
	3	5,780.58	5,444.81	12,080.27	0.00	0.00
Airport	2	84,534.87	84,534.87	8,186.64	0.00	0.00
	3	112,320.80	74,773.53	27,233.06	0.00	0.00
	4	105,113.57	84,331.25	178,480.77	31,799.22	-23.23
	5	105,570.96	73,825.11	593,612.00	-	-
	6	0.00	0.00	683,719.45	-	-100.00
	7	0.00	0.00	646,697.84	-	-100.0

Table 5.2: Results of Simulated Annealing with hybrid penalty

Both the algorithm based on the dynamic penalty method as well as the one based on the hybrid penalty method deliver excellent results up to $k = 5$. Actually for the sparsely populated network Small World both algorithms were able to detect the global optimum.

However, it is also noted that for $k = 6$ and $k = 7$, the number of state transitions L hat to be restricted to not exceed the maximum defined runtime of 10 minutes. Hence it follows that the algorithm can no longer converge in these cases. The solutions found remain infeasible (denoted by 0.00 fitness values).

5.1.1 Internal comparison

At first sight both methods seem to deliver similar results. To be able to identify which implementation performs better, a comparison of the ranks is carried out. For this purpose ten examples are drawn from every algorithm and sorted by their fitness value in ascending order (table 5.3).

After sorting the values the rank numbers are calculated. Therefore the ranks from 1 to 20 are summed up for each algorithm individually. If two or more values are identical, the corresponding ranks are summed up, and their arithmetic mean is calculated. This average is then added to the rank number.

Simulated Annealing - Rank Comparison ($k = 5$)						
Data set	Position	Value	Algorithm	Position	Value	Algorithm
BuddyPress	1	6721.15	H	11	6825.61	H
	2	6740.97	D	12	6833.11	H
	3	6759.50	D	13	6837.98	H
	4	6764.86	D	14	6838.23	D
	5	6765.38	D	15	6838.23	H
	6	6775.92	H	16	6838.80	H
	7	6786.18	H	17	6867.85	D
	8	6791.80	H	18	6867.85	D
	9	6797.32	H	19	6885.13	D
	10	6825.61	D	20	6909.53	D

Table 5.3: Simulated Annealing - Rank comparison

The rank number for the hybrid approach is calculated in equation 5.1, for the dynamic approach in equation 5.2. The rank numbers are $R_H = 97$ respectively. $R_D = 113$.

$$\begin{aligned} R_H &= 1 + 6 + 7 + 8 + 9 + \frac{10 + 11}{2} + 12 + 13 + \frac{14 + 15}{2} + 16 \\ R_H &= 97 \end{aligned} \tag{5.1}$$

$$\begin{aligned} R_D &= 2 + 3 + 4 + 5 + \frac{10 + 11}{2} + \frac{14 + 15}{2} + 17 + 18 + 19 + 20 \\ R_D &= 113 \end{aligned} \tag{5.2}$$

In accordance with the procedure described, all rank numbers are calculated. Unfortunately rank numbers might not always deliver a result that is easy to interpret. For example it cannot be expected that the difference between 106.0 and 104 (*BuddyPress*; it is $k = 2$) is significant (i.e the result based on a quality difference of the two variants); it is most likely just a random effect. Furthermore, a conclusion can only be drawn on the basis of significance tests. Therefore for all results the *Mann–Whitney–Wilcoxon* significance tests will be carried out.

The test works by simply calculating a statistic U_x for each method whose distribution under the null hypothesis H_0 is known. U_1 and U_2 are calculated as follows:

$$U_1 = R_1 - \frac{n_1(n_2 + 1)}{2} \tag{5.3}$$

and

$$U_2 = R_2 - \frac{n_2(n_2 + 1)}{2} \tag{5.4}$$

whereby R_1 and R_2 are summed up rank numbers and N_1/N_2 are the number of samples. For network *BuddyPress* with $k = 2$, U_1 and U_2 have been calculated in equation 5.5 and 5.6.

$$U_H = R_H - \frac{n_H(n_2 + 1)}{2} U_H = 106.0 - \frac{10(10 + 1)}{2} U_H = 51 \tag{5.5}$$

$$U_D = R_D - \frac{n_D(n_2 + 1)}{2} U_D = 104.0 - \frac{10(10 + 1)}{2} U_D = 49 \tag{5.6}$$

In the next step, the significance number is needed; it is taken from pre-calculated tables (attached in Appendix A). This significance number at a level of 5% for $n_1 = n_2 = 10$ is 23. If $Min(U_1, U_2)$ is smaller then the significance number, then the difference in the result is significant, otherwise not.

This test is carried out for all results, and in the rank comparison tables only significant results are printed in bold. All methods which could not deliver a feasible solution are stroked out.

Simulated Annealing - Comparison of rank numbers						
Data set	k	R_H	R_D	k	R_H	R_D
BuddyPress	2	106.0	104	5	97.0	113.0
	3	107.0	103	6	126.0	84.0
	4	102.0	108	7	115.0	~~95.0~~
Extended Model	2	99.0	111.0	5	92.0	118.0
	3	62.0	**148.0**	6	~~105.0~~	~~105.0~~
	4	101.5	108.5	7	~~105.0~~	~~105.0~~
High Density	2	110.0	100.0	5	128.0	82.0
	3	93.0	117.0	6	106.0	104.0
	4	92.5	117.5	7	113.0	97.0
Small World	2	105.0	105.0	3	110.0	100.0
Airport	2	120.0	90.0	5	96.0	114.0
	3	110.0	100.0	6	75.0	**135.0**
	4	112.0	98.0	7	~~105.0~~	~~105.0~~

Table 5.4: Simulated Annealing - Comparison of rank numbers

The choice of a superior implementation turned out to be tough. For the sparsely populated network *Small World* both approaches perform nearly equally. In the case of the network *Extended Model* and *Airport* the dynamic approach is slightly better. The same can be said for the network *BuddyPress*, whereas in this case the hybrid approach is performed slightly better. In the case of the *HighDensity* network, no significant method can be determined, but the hybrid method delivers the highest fitness values in 4 out of 7 cases.

Both approaches are chosen as a final algorithm : The hybrid method for the networks *HD* and *BuddyPress*, the dynamic method for the networks *EXT* and *Airport*.

5.2 Genetic Algorithm

The size of the populations and the number of generations were determined according to the proposed algorithm design in section 4.3.1. An overview can be found in table 5.5. The test cases for two the two proposed penalty methods and approaches were carried according to this setup.

The population for small sizes of k for the network *BuddyPress* is smaller than compared with the networks *EXT* or *HD*. This is due to the smaller number of vertices present in the network. The network *Small World* is very sparsely populated and therefore higher initial values are necessary.

Genetic Algorithm - Population size and number of generations						
		POP			GEN	
Data set	k	Population	Generations	k	Population	Generations
EXT, HD, Airport	2	80	14	2	20	500
	3	100	20	3	20	800
	4	120	20	4	20	1000
	5	200	40	5	40	2500
	6	220	40	6	40	2600
	7	240	40	7	40	2700
BuddyPress	2	80	14	2	20	300
	3	100	20	3	20	500
	4	120	20	4	20	800
	5	200	40	5	40	2,500
	6	220	40	6	40	2,600
	7	240	40	7	40	2,700
Small World	2	160	40	2	40	1,000
	3	200	40	3	40	1,200

Table 5.5: Size of population and number of generations

The results of the dynamic penalty method are presented in tables 5.6 and 5.8, those for the adaptive approach in tables 5.7 and 5.9. For each test case the algorithm was executed ten times. The tables include the results with the highest and lowest fitness, as well as the average runtime in milliseconds. Additionally, if an estimated global optimum is available, the deficit in absolute numbers and the percentage value are denoted.

Genetic Algorithm - Dynamic penalty (Population approach)						
Data set	k	Best	Weakest	Runtime ms	Deficit	% Deficit
BuddyPress	2	6,384.75	861.97	384.47	0.00	0.00
	3	6,605.48	5,831.28	611.24	0.00	0.00
	4	6,735.71	6,039.75	711.08	0.00	0.00
	5	6,909.53	0.00	1,516.24	-	-
	6	7,001.68	0.00	1,816.98	-	-
	7	7,001.68	0.00	1,828.40	-	-
Extended Model	2	24,631.71	1,359.63	409.92	0.00	0.00
	3	26,471.88	15,857.36	616,80	4,939.18	-15.72
	4	36,847.68	0.00	729.37	0.00	0.00
	5	36,847.68	0.00	729.37	-	-
	6	38,573.64	37,003.80	1935.67	-	-
	7	38,881.71	0,00	2237.94	-	-
High Density	2	923,11	917.87	385.81	0.00	0.00
	3	1,377,34	1,366.87	652.00	0.57	-0.04
	4	1,825.90	1,807.88	860.27	3.30	-0.18
	5	2,277.49	2,264.65	2,159.24	0.00	0.00
	6	2,717.16	2,702.37	2,789.27	-	-
	7	3,156.00	3,137.91	3,445. 46	-	-
Small World	2	6,225.39	0,.00	1,251.23	0.00	0.00
	3	0.00	0.00	2,113.23	-	-100.00
	4	*No clique available*				
Airport	2	84,534.87	30,642.70	384.75	0.00	0.00
	3	80,933.45	32,960.99	672.76	31,387.35	-27.94
	4	129,249.57	0.00	875.05	7,663.22	-5.60
	5	152,674.84	0.00	2,392.21		
	6	157,892.18	0.00	8,129.75	-	-
	7	172,372.88	0.00	4,326.84	-	-

Table 5.6: Results of Genetic Algorithm with dynamic penalty and POP

Genetic Algorithm - Adaptive penalty (Population approach)						
Data set	k	Best	Weakest	Runtime ms	Deficit	% Deficit
BuddyPress	2	6,384.75	1,111.65	400.33	0.00	0.00
	3	6,605.48	0.00	621.70	0.00	0.00
	4	6,675.20	0.00	704.25	11.89	-1.65
	5	6,885.13	0.00	1,538.89	-	-
	6	7,001.68	0.00	1,815.28	-	-
	7	6,935.87	0.00	1,828.01	-	-
Extended Model	2	24,631.71	1,609.45	396.56	0.00	0.00
	3	31,411.06	8,985.20	626.90	0.00	0.00
	4	36,847.68	0.00	712.23	0.00	0.00
	5	38,449.12	0.00	1,612.30	-	-
	6	38,573.64	0.00	1,823.03	-	-
	7	0.00	0.00	2,259.63	-	-100.00
High Density	2	923.11	915.95	502.06	0.00	0.00
	3	1,377.34	1,366.76	667.19	0.57	-0.04
	4	1,829.20	0.00	868.43	0.00	0.00
	5	2,270.15	0.00	2,235.22	7.33	-0.32
	6	2,713.67	0.00	2,841.00	-	-
	7	3,160.78	0.00	3,491.08	-	-
Small World	2	5,159.78	0.00	1,269.52	0.00	0.00
	3	0.00	0.00	2,219.93	-	-100.00
	4	*No clique available*				
Airport	2	83,883.26	35,010.47	377.81	651.61	-0.77
	3	109,933.15	54,086.76	657.46	2,387.65	-2.13
	4	108,981.39	0.00	890.98	27,931.40	-20.40
	5	152,674.84	0.00	2,398.33	-	-
	6	165,457.16	0.00	3,100.06	-	-
	7	173,168.22	0.00	4,001.62	-	-

Table 5.7: Results of Genetic Algorithm with adaptive penalty and POP

Genetic Algorithm - Dynamic penalty (Generation approach)						
Data set	k	Best	Weakest	Runtime ms	Deficit	% Deficit
BuddyPress	2	6,384.75	597.07	987.01	0.00	0.00
	3	6,605.48	1,051.64	1,326.19	0.00	0.00
	4	6,787.09	0.00	1,709.61	0.00	0.00
	5	6,909.53	0.00	5,479.23	-	-
	6	7,001.68	0.00	7,180.23	-	-
	7	7,001.68	0.00	7,461.74	-	-
Extended Model	2	24,631.71	1,326.12	1,000.42	0.00	0.00
	3	30,280.76	15,638.38	1,367.76	1,130.30	-3.60
	4	36,847.68	0.00	1,961.84	0.00	0.00
	5	38,449.12	0.00	5,352.35	-	-
	6	38,841.93	0.00	6,958.72	-	-
	7	38,881.71	0.00	9,790.18	-	-
High Density	2	923.11	917.42	983.47	0.00	0.00
	3	1,377.92	1,371.07	1,382.11	0.00	0.00
	4	1,829.16	1,821.55	2,072.82	0.00	0.00
	5	2,277.49	2,273.95	8,427.97	0.00	0.00
	6	2,720.64	2,703.49	11,289.27	-	-
	7	3,159.79	3,154.97	14,611.73	-	-
Small World	2	6,037.24	0.00	2,366.61	188.14	-3.02
	3	0.00	0.00	3,416.54	-	-100.00
	4	*No clique available*				
Airport	2	83,883.26	2,385.33	1,273.86	651.61	-0.77
	3	87,889.95	0.00	1,798.19	24,430.85	-21.75
	4	87,889.95	65,062.99	2,450.99	13,888.97	-10.14
	5	152,674.84	0.00	8,164.76	-	-
	6	162,221.53	0.00	10,432.10	-	-
	7	173,168.22	0.00	12,235.87	-	-

Table 5.8: Results of Genetic Algorithm with dynamic penalty and GEN

Genetic Algorithm - Adaptive penalty (Generation approach)						
Data set	k	Best	Weakest	Runtime ms	Deficit	% Deficit
BuddyPress	2	6,384.75	820.46	1,008.98	0.00	0.00
	3	6,605.48	1,039.37	1,318.86	0.00	0.00
	4	6,735.71	0.00	1,679.64	51.38	-0.76
	5	6,797.32	0.00	5,409.76	-	-
	6	6,953.24	0.00	6,745.84	-	-
	7	2,400.52	0.00	6,736.12	-	-
Extended Model	2	24,631.71	8,179.08	994.58	0.00	0.00
	3	31,411.06	16,107.38	1,313.05	0.00	0.00
	4	36,847.68	24,739.71	1,917.24	0.00	0.00
	5	38,449.12	0.00	5,239.51	-	-
	6	38,841.93	38,573.64	6,949.86	-	-
	7	0.00	0.00	7,684.23	-	-100.0
High Density	2	923.11	917.48	1,015.19	0.00	0.00
	3	1,377.92	1,369.12	1,414.47	0.00	0.00
	4	1,826.71	1,812.14	2,057.71	2.49	-0.14
	5	2,273.81	0.00	11,211.21	3.67	-0.16
	6	2,697.90	0.00	11,111.97	-	-
	7	3,154.00	0.00	17,244.98	-	-
Small World	2	4959,72	0.00	2,319.93	1265.67	-20.33
	3	0.00	0.00	3,412.54	-	-100.00
	4	*No clique available*				
Airport	2	84,534.87	11,752.74	1,297.76	0.00	0.00
	3	102,918.79	0.00	1,805.37	9,402.01	-8.37
	4	118,340.12	79,047.57	2,529.55	18,572.67	-13.57
	5	152,674.84	0.00	8,214.34	-	-
	6	165,457.16	0.00	10,159.69	-	-
	7	174,731.23	0.00	11,766.19	-	-

Table 5.9: Results of Genetic Algorithm with adaptive penalty and GEN

At first sight, four execution configurations do not to seem to differ by any great dimensions. For all of them it was proven that there are occasions where the global optimum can only be approximated but not exactly determined.

Worth highlighting is the fact thand methods in the case of network *Small World* delivered extremely poor results. None of them was able to identify a feasible solution where $k = 3$.

Particular notice also deserves the outlier of the *Extended Model* where $k = 3$. No configuration including the dynamic penalty method could find a global optimum, whereas the adaptive penalty performed much better.

5.2.1 Internal comparison

To be able to identify possible superior implementations again a further comparison of the ranks was carried out. Since two different approaches are used, their their different penalty methods are first checked against each other. The results can be seen in table 5.10 for the population approach and in table 5.11 for the generation approach.

Genetic Algorithm - Comparison of rank numbers (POP)						
Data set	k	R_A	R_D	k	R_A	R_D
BuddyPress	2	106.5	103.5	5	91.5	118.5
	3	100.0	110.0	6	98.5	111.5
	4	92.5	117.5	7	102.5	107.5
Extended Model	2	109.5	100.5	5	**133.0**	77.0
	3	117.5	92.5	6	98.0	112.0
	4	130.0	80.0	7	~~90.0~~	120.0
High Density	2	104.5	105.5	5	70.0	**140.0**
	3	115.0	95.0	6	82.0	128.0
	4	83.0	127.0	7	80.0	130.0
Small World	2	61.5	**148.5**	3	~~105.0~~	~~105.0~~
Airport	2	118.0	92.0	5	113.5	96.5
	3	115.0	95.0	6	122.0	73.0
	4	99.0	110.5	7	107.0	103.0

Table 5.10: Genetic Algorithm - Comparison of rank numbers (POP)

When considering significance levels for the networks *HighDensity* and *SmallWorld*, the differences are marginal. Solely the dynamic approach was able to out-perform the adaptive method once. For the networks *Airport* and *BuddyPress* no difference could be observed between the two penalty methods. Obviously the differences are only marginal, but a decision has to be made. Therefore the dynamic penalty method was selected as the superior implementation in combination with the population approach.

Genetic Algorithm - Comparison of rank numbers (GEN)						
Data set	k	R_A	R_D	k	R_A	R_D
BuddyPress	2	115.0	95.0	5	74.5	**135.5**
	3	105.0	105.0	6	90.0	120.0
	4	92.0	118.0	7	91.0	119.0
Extended Model	2	105.5	104.5	5	123.5	86.5
	3	105.0	105.0	6	**149.5**	60.5
	4	110.0	100.0	7	~~100.0~~	110.0
High Density	2	117.0	93.0	5	55.0	**155.0**
	3	96.5	113.5	6	55.0	**155.0**
	4	66.0	**144.0**	7	55.0	**155.0**
Small World	2	71.0	**139.0**	3	~~105.0~~	~~105.0~~
Airport	2	116.0	94.0	5	108.0	102.0
	3	102.0	108.0	6	94.5	115.5
	4	114.0	96.0	7	113.0	97.0

Table 5.11: Genetic Algorithm - Comparison of rank numbers (GEN)

The results for the generation approach are easier to interpret. Indeed the dynamic method is able to identify slightly better solutions for the test networks *BuddyPress* and *Small World*. However, for the network *High Density* the approach is obviously superior: it outperformed the adaptive penalty method in 4 out of 6 cases. No significant difference between the two penalty variants could be observed within the test network *Airport*. Again the adaptive penalty method delivered better results for the *Extended Model* network, though the approach was not able to find a feasible solution for $k = 7$. For all these reasons, here too the dynamic penalty method was selected as superior.

In the second step of the internal comparison, the two selected superior approaches are used and compared again. The full data of that step can be seen in table 5.12.

Genetic Algorithm - Comparison of rank numbers (Best 2)						
Data set	k	R_{Gen}	R_{Pop}	k	R_{Gen}	R_{Pop}
BuddyPress	2	90.5	119.5	5	117.5	92.5
	3	117.0	93.0	6	112.5	97.5
	4	88.0	122.0	7	114.0	96.0
Extended Model	2	92.5	117.5	5	**145.0**	64.5
	3	105.0	105.0	6	65.0	**145.0**
	4	120.5	89.5	7	96.0	114.0
High Density	2	89.0	121.0	5	**145.5**	64.5
	3	**135.5**	74.5	6	**145.0**	65.0
	4	**144.0**	66.0	7	**154.0**	56.0
Small World	2	75.5	**134.5**	3	~~105.0~~	~~105.0~~
Airport	2	103.0	107.0	5	106.5	103.5
	3	78.0	132.0	6	103.0	107.0
	4	110.0	100.0	7	103.5	106.5

Table 5.12: Genetic Algorithm - Comparison of rank numbers (Best 2)

For the network *BuddyPress* no difference in the quality of results can be detected. Without question, the generation approach delivers far better results than the population approach when it comes to network *HighDensity*. Regarding network *ExtendedModel*, no significant differences seemed to exist at first sight, as both approaches outperformed the other one time. But upon having a look at the actual fitness values (GBC), it was observed that the generation approach delivered the highest values for all sizes of k. For the network *Small World* none of the proposed genetic algorithms delivered any good results. For all these reasons, the generation approach with the dynamic penalty method was the final choice.

6 Conclusio

6.1 Genetic Algorithm versus Simulated Annealing

This chapter now compares the results of Simulated Annealing and the Genetic Algorithm with consideration of runtime and quality.

Table 6.1 compares the rank values of the superior algorithms. The newly introduced column *Top* specifies the algorithm with the best result (highest fitness value).

Comparison of Rank numbers (SA vs. GA)								
Data set	k	R_{GA}	R_{SA}	Top	k	R_{GA}	R_{SA}	Top
BuddyPress	2	74.0	**136.0**	Both	5	119.0	91.0	GA
	3	87.0	123.0	Both	6	105.0	105.0	GA
	4	80.5	129.5	Both	7	115.0	95.0	GA
Extended Model	2	63.0	**147.0**	Both	5	**145.0**	65.0	GA
	3	83.0	127.0	SA	6	110.0	~~100.0~~	GA
	4	**135.0**	75.0	GA	7	110.0	~~100.0~~	GA
High Density	2	66.0	**144.0**	Both	5	**155.0**	55.0	GA
	3	99.0	111.0	GA	6	**145.0**	65.0	GA
	4	129.5	80.5	GA	7	**155.0**	55.0	GA
Small World	2	55.0	**155.0**	SA	3	~~55.0~~	**155.0**	SA
Airport	2	55.0	**155.0**	SA	5	**141.0**	69.0	GA
	3	59.0	**151.0**	SA	6	**145.0**	~~65.0~~	GA
	4	102.0	108.0	GA	7	125.0	~~85.0~~	GA

Table 6.1: Comparison of rank numbers (SA vs. GA)

At first sight it seems that both implemented algorithms are equal, as they both outperformed each other seven times. But when taking the highest fitness values (GBC values) into account, a different conclusion can be drawn. Excluding those

cases where the highest fitness value is equal, the Genetic Algorithm delivers the highest fitness value for a configuration in 17 out of 21 cases (a single configuration is a network with a single size of k). For a meta heuristic a higher fitness value is more important than better average results because their aim is to approximate to the optimum solution.

Additionally, the Genetic Algorithm has the highest fitness value for all sizes of k higher than 4. Furthermore, starting at sizes of $k = 5$, the Simulated Annealing algorithms started to generating only infeasible solutions, and was only able to outperform the Genetic Algorithm in 1 of 13 cases.

6.2 Data interpretation

The real world data set of the Top 500 US airports were used to construe the results of the MGBC cliques. First of all it has to be noted that the data set only includes domestic flights within the USA and does not consider cargo flights. The data is taken from 2006.

The optimum solutions found by all algorithms of the airport network can be seen in table 6.2. An interesting observation is that only seven vertices occur in all six solutions together.

Best results for airport network			
	k	Fitnessvalue (GBC)	Result Vertices
Airport	2	84,534.87	[7, 56]
	3	112,320.80	[7, 14, 56]
	4	136,912.79	[6, 7, 14, 56]
	5	152,674.84	[3, 6, 7, 14, 56]
	6	165,457.16	[1, 3, 6, 7, 14, 56]
	7	174,731.23	[1, 2, 3, 6, 7, 14, 56]

Table 6.2: Best results for airport network

Figure 6.1 shows the airports (vertice numbers in parantheses) on the US American map. It can be seen that they are spread across over the whole country and not accumulated in a specific area. However, with the exception of San Francisco, no coastal towns are represented within the solutions.

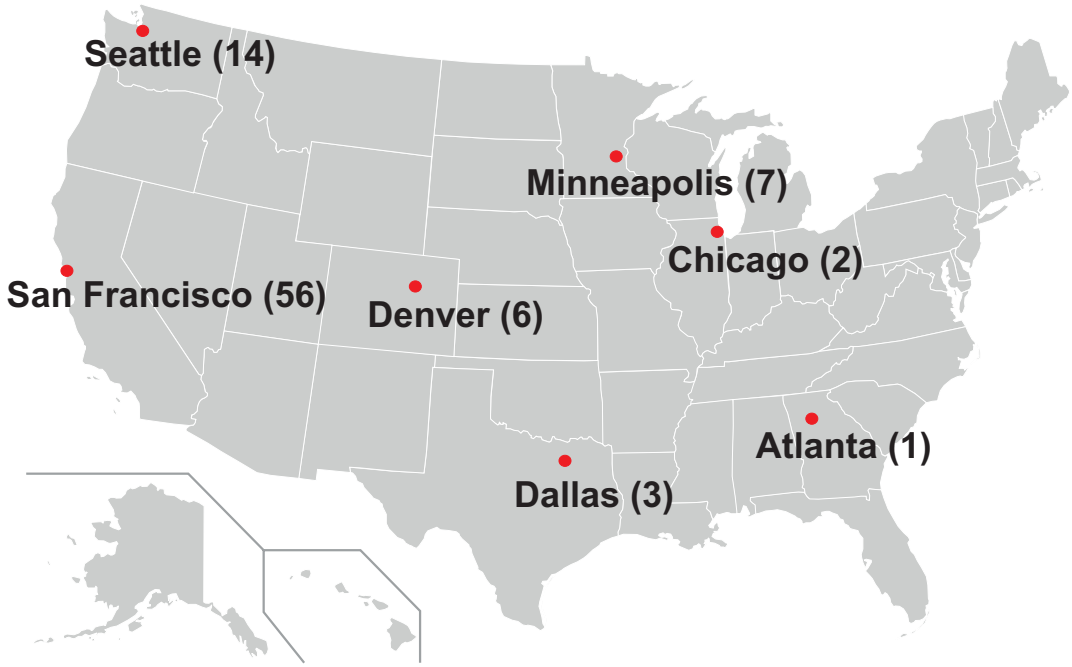

Figure 6.1: Airport network - $k = 7$ clique

The question that arises is if there is any connection between the solution airports and any other data such as population. The *Federal Aviation Administration* (FAA) publishes an annual list of the most frequented airports (international and domestic flights) in the USA. The list for 2006 can be seen in the table below [13]:

Commercial Service Airports in US - Passenger Volume				
	Rank	Airport City	Rank	Airport City
Airport	1	Atlanta	11	Detroit
	2	Chicago	12	Minneapolis
	3	Los Angeles	13	Orlando
	4	Dallas	14	San Francisco
	5	Denver	15	Miami
	6	Las Vegas	16	Philadelphia
	7	New York/JFK	17	Charlotte
	8	Phoenix	18	Seattle
	9	Houston	19	Boston
	10	Newark	20	Washington

Table 6.3: Commercial service airports in US, ranked by passenger volume

The seven airports of the best found clique also have the following characteristics:

- **Seattle (14)** was the 18th most frequented airport in the USA in 2006 and served as major airline hub for *Alaska Airlines* and *Horizon Air*.
- **San Francisco (56)** was the 14th most frequented airport of the USA in 2006 and served as major airline hub for *United Airlines* and *Virgin America*.
- **Denver (6)** was the 5th most frequented airport in the USA in 2006 and served as major airline hub for *United Airlines*, *Horizon Air*, *Great Lakes Airlines* and *Frontier Airlines*.
- **Minneapolis (7)** was the 12th most frequented airport of the USA in 2006 and served as major airline hub for *Delta Air Lines*, *Great Lakes Airlines* and *Sun Country Airlines*.
- **Dallas (3)** was the 4th biggest metropolitan area and the most frequented airport in the USA in 2006. The city served as major airline hub for *American Airlines*.
- **Atlanta (1)** was the 9th biggest metropolitan area and the 5th most frequented airport in the USA in 2006. The city served as major airline hub for *Delta Air Lines*.
- **Chicago (2)** was the 3rd biggest metropolitan area and the 2nd most frequented airport in the USA in 2006. The city served as major airline hub for *American Airlines* and *United Airlines*.

For further investigation, a new data set consisting of all airports, four characteristics and a variable as whether an airport was contained in a solution, was created. The four characteristics describe:

- **METRO10** - Was the airport within the 10 biggest metropolitan areas within the USA?
- **TOP10** - Was the airport one of the 10 most frequented airports of the USA in 2006 (including domestic and international flights)?
- **TOP20** - Was the airport one of the 20 most frequented airports of the USA in 2006 (including domestic and international flights)?
- **HUB5** - Was the airport an airline hub of one of the five biggest airlines (*Delta Air Lines*, *United Airlines*, *Southwest Airlines*, *American Airlines*, *US Airways*) in the USA?

First a linear model depending on all variables was created from this data set, and from that the t-test carried out. At a significance level of 5% all variables except HUB5 are significant.

The correlation between the characteristics and the solution variable are

$$\begin{aligned} Corr_{HUB5} &= 0.47042 \\ Corr_{TOP10} &= 0.494588 \\ Corr_{TOP20} &= 0.5982009 \\ Corr_{METRO10} &= 0.3458007 \end{aligned} \tag{6.1}$$

Correlation is a statistical relationship between two variables and always a value between -1.0 and $+1.0$. If the value tends toward 0, the connection between the two variables is small. Therefore it can be said that there is medium to high connection between an airport being one of the 20 most frequented airports in the USA and representing a vertice in a solution. Also 0.47042 can be seen as a high value, because there are only 7 airports in the solution column but 24 airports defined as hubs.

Other highly frequented airports like New York (No.7 for passenger volume in 2006 in the USA) or Las Vegas (No.6 for passenger volume in 2006 in the USA) are either internationally better known and more frequented or simply not that well served with domestic flights (in most cases no hub for an airline).

6.3 Final remarks

In the previous five chapters, this thesis provided an an introduction to Social Network Analysis and Group Betweenness Centrality and carried out a detailed analysis of different algorithms with several approaches and methods.

At the end of this Master's Thesis following conclusions can be drawn:

- All tests were carried out on an *Intel Core i5 CPU*. When using a different CPU, there is an - albeit unlikely - possibility that algorithms may perform highly contradictorily due to different design and implementation of the CPU.
- As long as they do not represent extreme examples, there is no obvious difference regarding the quality of results between synthetic and real world data.
- In the case of 4 out of 5 networks the optimum solution with size $k-1$ was always part of the solution with size k. This can be seen in table 4.1 of chapter 4. This seems to be a quite widespread behaviour within networks. The only network where this relation does not exist is within the extreme network *Small World*.

- As a test case, the penalty value for the Simulated Annealing approach with the dynamic penalty method was increased ($\alpha = 2$ instead of 1) to clarify whether feasible solutions could be found with that higher penalty value. As a test case this modification was carried out with the network EXT and $k = 6$. However, this newly adapted configuration was not able to find any feasible solution candidate.
- The computing time needed by Simulated Annealing until it converges is compared to the Genetic Algorithm extensive and several times higher. This supposition was also made by Henderson, Jacobson and Johnson in [17].
- If given enough computing time, Simulated Annealing is then likely to find a feasible solution. It could be said that SA is more reliable than the GA. In addition Simulated Annealing requires significantly less effort in parametrisation.
- Regarding the network *SmallWorld* for $k = 3$, SA outperforms GA in all cases with all approaches.
- No significant difference between the two penalty variants of Simulated Annealing were be observed.
- The choice of the inital temperature based on the research of *Thompson* and *Bilbro* as described in section 4.2.1 may seem peculiar in the beginning, because the number of edges and vertices (viz. the size of the network) only influences the T_0 indirectly. However, after all tests were carried out, it proved itself as a good approximation.
- The Genetic Algorithm needs more executions until it is able to detect an excellent or even a feasible solution. GAs can definitely can find global optima, but there are many other attraction poles to which they may converge. The more complex the network is, the more executions the GA requires. This behaviour was also described by [29].
- Within Genetic Algorithms diversity maintenance should be of primary concern. Therefore a mutation rate of 1% was used, which may seem quite high, but in relation to complex networks this is necessary.
- In 17 of our test instances regarding the highest fitness value, GA outperformed SA, whereas SA outperformed GA only in 5 instances. This difference increases as the size of k rises. Above $k = 4$, GA outperforms SA in 12 of 13 cases.
- In the airport network, a medium to high correlation could be observed between an airport being a hub for one of the five biggest airlines in the USA and being a vertice in a solution. This correlation also existed for an airport being one of the 20 most frequented airport in the USA.

Bibliography

[1] Albert, R., and Barabasi, A.-L. Topology of evolving networks: local events and universality. Physical Review Letters 85 (2000), 5234.

[2] Barrat, A., Barthélemy, M., Pastor-Satorras, R., and Vespignani, A. The architecture of complex weighted networks. Proceedings of the National Academy of Sciences of the United States of America 101, 11 (2004), 3747–3752.

[3] Batagelj, V., and Brandes, U. Efficient generation of large random networks. OFFEN 71, 3 (March 2005).

[4] Batagelj, V., and Mrvar, A. Pajek - Reference Manual, 2.05 ed. Ljubljana, Slovenia, September 2011.

[5] Ben-Ameur, W. Computing the initial temperature of simulated annealing. Computational Optimization and Applications 29 (2004), 369–385.

[6] Bollobás, B. Random Graphs. Cambridge University Press, Cambridge, UK, 2001.

[7] Borgatti, S. P. Identifying sets of key players in a social network. Comput. Math. Organ. Theory 12, 1 (April 2006), 21–34.

[8] Borgatti, S. P., Mehra, A., Brass, D. J., and Labianca, G. Network analysis in the social sciences. Science 323 (2009), 892—895.

[9] Choudhury, T. K. Sensing and Modeling Human Networks. PhD thesis, Massachusetts Institute of Technology, 2004.

[10] De Jong, K. A. An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan, 1975.

[11] Eshelman, L. J., Caruana, R., and Schaffer, J. D. Biases in the crossover landscape. In ICGA (1989), pp. 10–19.

[12] Everett, M., and Borgatti, S. P. The centrality of groups and classes. Journal of Mathematical Sociology 23 (1999), 181–201.

[13] Federal Aviation Administration. Calendar year 2006 passenger activity - commercial service airports in us, October 2007.

[14] Fink, M., and Spoerhase, J. Maximum betweenness centrality: Approximability and tractable cases. CoRR abs/1008.3503 (2010).

[15] Freeman, L. C. The Development of Social Network Analysis - A Study in the Sociology of Science. Empirical Press, 2004.

[16] Hadj-Alouane, A. B., and Bean, J. C. A genetic algorithm for the multiple-choice integer program. Tech. rep., University of Michigan, Ann Arbour, MI, USA, September 1993.

[17] Henderson, D., Jacobson, S. H., and Johnson, A. W. The theory and practice of simulated annealing. In Handbook of Metaheuristics. Kluwer Academic Publishers, New York, USA, 2003.

[18] Holland, J. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI, 1975.

[19] Jamali, M., and Abolhassani, H. Different aspects of social network analysis. In Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (Washington, DC, USA, 2006), IEEE Computer Society, pp. 66–72.

[20] Joines, J., and Houck, C. R. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. In Proceedings of the First IEEE Conference on Evolutionary Computation (1994), IEEE Press, pp. 579–584.

[21] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. Science 220 (1983), 671—680.

[22] Königsberg bridge problem. Encyclopædia Britannica Online. Encyclopædia Britannica Inc., 2012. Web. 23 Apr. 2012.

[23] Levine, D. Genetic algorithms: A practitioner's view. INFORMS Journal on Computing (1997), 256–259.

[24] Martín, J. F. D., and Sierra, J. R. A comparison of cooling schedules for simulated annealing. In Encyclopedia of Artificial Intelligence. IGI Global, 2009, pp. 344–352.

[25] Mitchell, M. An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA, USA, 1998.

[26] Puzis, R., Dolev, S., and Elovici, Y. Routing betweenness centrality. J. ACM 57, 4.

[27] Puzis, R., Dolev, S., and Elovici, Y. Fast algorithm for successive computation of group betweenness centrality. Physial Review E 76 (November 2007), 056709.

[28] Puzis, R., Dolev, S., Elovici, Y., and Zilberman, P. Incremental deployment of network monitors based on group betweenness centrality. Information Processing Letters 109 (September 2009), 1172—-1176.

[29] Reeves, C. Genetic algorithms. In Handbook of Metaheuristics. Kluwer Academic Publishers, New York, USA, 2003.

[30] Shojaee, K., T, M. B., and G, H. S. Investigation on the choice of the initial temperature in the simulated annealing: A mushy state sa for tsp. In Proceedings of the 2009 17th Mediterranean Conference on Control and Automation (2009), pp. 1050–1055.

[31] Smith, A., Smith, A. E., Coit, D., Baeck, T., Fogel, D., and Michalewicz, Z. Penalty functions, 1997.

[32] Strenski, P., and Kirkpatrick, S. Analysis of finite length annealing schedules. Algorithmica 6 (1991), 346–366.

[33] Thompson, D. R., and Bilbro, G. L. Sample-sort simulated annealing. IEEE Transactions on Systems, Man, and Cybernetics, Part B 35, 3 (2005), 625–632.

[34] Černý, V. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications 45 (1985), 41—-51.

[35] Wasserman, S., and Faust, K. Social Network Analysis - Methods and Applications. Camebridge University Press, 1994.

[36] Özgür Yeniay. Penalty function methods for constrained optimization with genetic algorithms. Mathematical and Computational Applications 10 (2005), 45–56.

List of Abbreviations

BC	Betweenness Centrality
FAA	Federal Aviation Administration
GA	Genetic Algorithm
GBC	Group Betweenness Centrality
MBC	Maximum Betweenness Centrality
MGBC	Maximum Group Betweenness Centrality
SA	Simulated Annealing
SNA	Social Network Analysis

Appendix A

Critical Values for the Mann-Whitney U-Test

Level of significance: 5% (P = 0.05)

Size of the largest sample (n_2)

Size of the smallest sample (n_1)	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
3	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11	11	12	13	13
4	1	2	3	4	4	5	6	7	8	9	10	11	11	12	13	14	15	16	17	17	18	19	20	21	22	23
5	2	3	5	6	7	8	9	11	12	13	14	15	17	18	19	20	22	23	24	25	27	28	29	30	32	33
6		5	6	8	10	11	13	14	16	17	19	21	22	24	25	27	29	30	32	33	35	37	38	40	42	43
7			8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48	50	52	54
8				13	15	17	19	22	24	26	29	31	34	36	38	41	43	45	48	50	53	55	57	60	62	65
9					17	20	23	26	28	31	34	37	39	42	45	48	50	53	56	59	62	64	67	70	73	76
10						23	26	29	33	36	39	42	45	48	52	55	58	61	64	67	71	74	77	80	83	87
11							30	33	37	40	44	47	51	55	58	62	65	69	73	76	80	83	87	90	94	98
12								37	41	45	49	53	57	61	65	69	73	77	81	85	89	93	97	101	105	109
13									45	50	54	59	63	67	72	76	80	85	89	94	98	102	107	111	116	120
14										55	59	64	67	74	78	83	88	93	98	102	107	112	118	122	127	131
15											64	70	75	80	85	90	96	101	106	111	117	122	125	132	138	143
16												75	81	86	92	98	103	109	115	120	126	132	138	143	149	154
17													87	93	99	105	111	117	123	129	135	141	147	154	160	166
18														99	106	112	119	125	132	138	145	151	158	164	171	177
19															113	119	126	133	140	147	154	161	168	175	182	189
20																127	134	141	149	156	163	171	178	186	193	200
21																	142	150	157	165	173	181	188	196	204	212
22																		158	166	174	182	191	199	207	215	223
23																			175	183	192	200	209	218	226	235
24																				192	201	210	219	228	238	247
25																					211	220	230	239	249	258
26																						230	240	250	260	270
27																							250	261	271	282
28																								272	282	293
29																									294	305
30																										317

Zusammenfassung

In der Analyse sozialer Netzwerke ist der Begriff der *Gruppen-Betweenness* eine Einheit, welche den Einfluss einer Gruppe innerhalb eines Netzwerks misst. Die Gruppen-Betweenness einer Teilmenge von Individuen in einem sozialen Netzwerk ist umso größer, je mehr kürzeste Pfade zwischen Paaren von anderen Personen im Netzwerk über Mitglieder der betrachteten Teilmenge verlaufen.

Es gibt Algorithmen zur Bestimmung der Gruppen-Betweenness, und auch das Problem der Bestimmung einer Teilmenge gegebener Größe mit maximaler Gruppen-Betweenness wurde in der Literatur bereits behandelt. Das Ziel dieser Arbeit ist aber nicht nur eine Gruppe mit maximalen Gruppen-Betweenness Wert zu finden, sondern auch einen Algorithmus für die Suche nach Gruppen, in der jedes Mitglied mit jedem anderen verbunden ist (sogennante Cliquen), zu entwickeln.

Da das Problem np-schwer ist, ist der entwickelte Algorithmus von metaheuristischer Natur. Zur Qualitätssicherung wurde nicht nur ein Algorithmus angewendet, sondern zwei unterschiedliche Techniken - *Simulated Annealing* (SA) und *Genetischer Algorithmus* (GA) - implementiert und verglichen.

Im Zusammenhang mit dieser Problemstellung sind die besten Teilmengen eines Netzwerks nicht zulässig, das heißt sie stellen keine Clique dar. Daher wird ein geeigneter Ansatz benötigt um entweder unzulässige Lösungen wieder auszuscheiden (*Penalty Methode*) oder aber nur zulässige Lösungen zu generieren (*Repair Methode*). Die hier vorgestellten Algorithmen basieren auf zwei verschiedenen Penalty-Methoden.

Die der Analyse zugrunde liegenden Daten sind sowohl von realweltlichen sozialen Netzwerken als auch von synthetisch erzeugten Daten abgeleitet.

Abstract

In *Social Network Analysis* (SNA) the concept of *Group Betweenness Centrality* (GBC) is a unit to measure the influence of a group within a network. It is defined as the more shortest paths of the network pass through a subset of individuals, the greater the betweenness of this subset of individuals is.

There are algorithms for determining the Group Betweenness Centrality and also for determining a subset of given size with maximum GBC. However, the aim of this thesis is not only to find a group with maximum GBC, but also to develop an algorithm for finding a group in which every member is connected to every other member (also called clique).

As the problem itself is np-hard the proposed algorithm is of a meta heuristic nature. For quality assessment not only one algorithm was implemented, instead the two techniques *Simulated Annealing* (SA) and *Genetic Algorithm* (GA) were applied and compared.

Since most of the generated solutions are not feasible in the context of this problem statement, i.e. don't compose a clique, a suitable approach to either eliminate unacceptable solutions (*penalty-function method*) or only generate feasible solutions (*repair function method*), is required. The final algorithms work with two different penalty method approaches.

The underlying data used in the analysis is derived from real-world data sets of social networks as well as from synthetically generated data.

Curriculum Vitae

Marlene Weiss

Geboren am: 28. Februar 1988
Geboren in: St. Pölten, Österreich

Ausbildung

Dipl.-Ing.
Wirtschaftsinformatik (Universität Wien) 10/2010 - 10/2012

B.Sc.
Computer- und Mediensicherheit (FH Hagenberg) 10/2007 - 07/2010